非構造CFD解析ツールの構築と有用性検証

非構造 CFD 解析ツールの構築と有用性検証

課 題 番 号 09555059

１９９７年（平成９年）度～１９９９年（平成１１年）度
科学研究費補助金（基盤研究(B)(２)）

研究成果報告書

２０００年３月

研究代表者　中橋 和博

（東北大学大学院工学研究科航空宇宙工学専攻・教授）

研究課題　　非構造 CFD 解析ツールの構築と有用性検証

課題番号　　09555059

研究組織	研究代表者	中橋　和博	(東北大学大学院工学研究科・教授)
	研究分担者	福西 祐	(東北大学大学院工学研究科・助教授)
		大林　茂	(東北大学大学院工学研究科・助教授)
		加藤　琢真	(東北大学大学院工学研究科・助手)
		安藤　安則	(石川島播磨重工業(株)・技術研究所・流体・燃焼部・課長（研究職))
		三谷　徹	(航空宇宙技術研究所角田ロケット開発センターラム燃焼研究室・室長（研究職))

研究経費	１９９７年度	５，９００千円
	１９９８年度	４，４００千円
	１９９９年度	１，９００千円
	合計	１２，２００千円

研究発表

(1) 学術誌

1. K. Nakahashi and E. Saitoh, “Space-Marching Method on Unstructured Grid for Supersonic Flows with Embeded Subonic Regions”, AIAA Journal, Vol.35, No.8, pp.1280-1285, August 1997.
2. 小寺正敏，中橋和博，大林茂，苅田丈士，三谷徹，“スクラムジェットインレット内における境界層流入の影響,”日本航空宇宙学会誌, Vol. 45, No. 519, 平成９年４月, pp. 216-221.
3. D. Sharov, K. Nakahashi, “Reordering of Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations”, AIAA Journal, Vol.36, No.3, pp.484-486, 1998.
4. D. Sharov, K. Nakahashi, “Hybrid Prismatic/Tetrahedral Grid Generation for Viscous Flow Applications”, AIAA Journal, Vol.36, No.2, pp.157-162, 1998.
5. 小寺正敏，中橋和博，Dmitri Sharov，“ハイブリッド非構造格子法による３次元衝撃波／乱流境界層干渉の数値シミュレーション，” 日本機械学会論文集(B 編)，64 巻，627 号，pp.3669-3674, 1998 年 11 月.
6. K. Nakahashi, D. Sharov, S. Kano, M. Kodera, “Applications of Unstructured Hybrid Grid Method to High-Reynolds Number Viscous Flows ”, International Journal for Numerical Methods in Fluids, Vol.31, pp.97-111, 1999.

7. S. Kano and K. Nakahashi,, "Flow Computations around Delta Wings Using Unstructured Hybrid Grids", Journal of Aircraft, Vol.36, No.2, pp.374-379, 1999.
8. 小寺正敏，中橋和博，五十嵐康隆，苅田丈士，平岩徹夫，三谷徹，"ハイブリッド非構造格子法を用いたスクラムジェット内部流の数値解析"，日本機会学会論文集（B編）６５巻，６３３号, pp.1513-1519, 1999.
9. T. Mitani, T.Kanda, T. Hiraiwa, Y. Igarashi, K. Nakahashi, "Drags in Scramjet Engine Testing: Experimental and Computational Fluid Dynamics Studies", Journal of Propulsion and Power, July-August 1999. Vol.15, No.4
10. K. Nakahashi, D. Sharov, S. Kano, M. Kodera, ""High-Reynolds number viscous flow computations by unstructured hybrid grid method", Computational Fluid Dynamics Journal, Vol.8, No.2, pp.232-242, 1999.
11. 加藤，中橋：非構造格子上での擬似圧縮性法を用いた非圧縮流れ解析の精度検証，日本機械学会論文集 B 編，65 巻 640 号（1999），pp. 3899-3905.
12. 加藤，村山，伊藤，中橋：非構造格子上における非定常非圧縮流れの計算日本機械学会論文集 B 編，66 巻 641 号（2000），pp. 4-5.
13. 伊藤 靖，中橋 和博, "GUI を用いた非構造表面格子生成法"，日本航空宇宙学会誌，Vol. 48, No. 554 (3 月号)，2000.
14. 富樫史弥、中橋和博, "非構造オーバーセット格子による超音速実験機・ブースター分離シミュレーション"，日本航空宇宙学会誌，Vol. 48, No. 556 (5 月号)，2000.

(2) 国際会議発表

1. D. Sharov, K. Nakahashi, "Reordering of Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations", Proc. 13th AIAA CFD Conf., AIAA-97-2102-CP, Vol.CP979, (1997), pp.131-138.
2. N. Okamoto, K. Nakahashi, S. Obayashi, "Unstructured Multigrid Method for Euler Equations", Proc. Int. Conf. on Fluid Eng., JSME Centennial Grand Congress, Tokyo, Vol.2 (1997), pp.801-805.
3. Kazuhiro Nakahashi and Dmitri Sharov, "Applications of Unstructured Grid Method to High-Reunolds Number Viscous Flows", International Conference on Finite Elements in Fluids, January 5-8, 1998 at Tucson.
4. Kodera, M., Nakahashi, K., Hiraiwa, T., Kanda, T., and Mitani, T., ""Scramjet Inlet Flow Computations by Hybrid Grid Method," AIAA Paper 98-0962, 1998.
5. Sharov, D., Nakahashi, K., "Low Speed Preconditioning and LU-SGS Scheme for 3-D Viscous Flow Computations on Unstructured Grids", AIAA Paper 98-0614, 1998.
6. Y. Igarashi, K. Nakahashi, M. Kodera, T. Mitani, T. shimura, ""Experimental and Numerical Analysis of Scramjet Internal Flows -Comparative Studies of Engine Drag-", AIAA 98-1512, April 1998.
7. K. Nakahashi, D. Sharov, T. Katoh and M. Kodera, "On Capability of Unstructured Hybrid Grid Method", 16th International Conference of Numerical Method in Fluid, Arcachon, FRANCE, July

6-10, 1998.

8. Obayashi, S., Nakahashi, K., Oyama, A. and Yoshino, N., "Design Optimization of Supersonic Wings Using Evolutionary Algorithms," Proceedings of the Fourth ECCOMAS Computational Fluid Dynamics Conference, Athens, John Wiley & Sons Ltd, Chichester, 1998, pp. 575-579.
9. D. Sharov, and K. Nakahashi, ""Curvature Adapted Triangulation of Surface Models via Incremental Insertion Algorithm", Proc. of 6th International Conference on Numerical Grid Generation in Computational Field Simulations, pp.695-704, 1998.
10. Yasutaka Igarashi, Masatoshi Kodera, Kazuhiro Nakahashi and Tohru Mitani, "Computation of cold flows inside of a scramjet engine using unstructured grid" 1st Eastern-Western High Speed Flow Fields Conference and Workshop, Kyoto, Nov. 1998.
11. H. Morino, K. Nakahashi, "Space-Marching Method on Unstructured Hybrid Grid for Supersonic Viscous Flows", AIAA Paper 99-0661, 37th Aerospace Sciences Meeting and Exhibit, Reno, January 1999.
12. Kodera, M., and Nakahashi, K, "Extension of Unstructured Hybrid Grid Method to Supersonic Combustion Flows" AIAA Paper 99-0486, 37th Aerospace Sciences Meeting and Exhibit, Reno, January 1999.
13. K. Nakahashi, F. Togashi, D. Sharov, "An Intergrid-Boundary Definition Method for Overset Unstructured Grid Approach", Proc. 14th AIAA Computational Fluid Dynamics Conference, AIAA-99-3304, 1999.
14. Kazuhiro Nakahashi Fumiya Togashi, Dmitri Sharov, "On the Use of Overset Grid Concept for Unstructured Grid Approach", Proc. International Symposium on Computational Fluid Dynamics, Bremen, Germany, Sep.1999.
15. Kodera, M., Sunami, T., Nakahashi, K., "Numerical Analysis of SCRAMHET Combusting Flows by Unstructured Hybrid Grid Method", AIAA 2000-0886, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.
16. Ito, Y., Nakahashi, K., "Direct Surface Triangulation Using Stereolithography (STL) Data", AIAA-2000-0924, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.
17. Murayama, M., Nakahashi, K., Sawada, K., "Numerical Simulation of Vortex Breakdown Using Adaptive Refinement with Vortex-Center Identification", AIAA-2000-0806, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.
18. Togashi, F., Nakahashi, K., Ito, Y., Iwamiya, T., Shimbo, Y., "Flow Simulation of NAL Experimental Supersinic Airplane/Booster Separation Using Overset Unstructured Grids", AIAA-2000-1007, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.
19. H-J. Kim, S. Obayasshi, K. Nakahashi, "Aerodynamic Optimization of Supersonic Wing-Nacelle Configuration Using an Adjoint Method with the Unstructured-Grid Approach", International Workshop on Numerical Simulation Technology for Design of Next Generation Supersonic Civil Transport, Tokyo, Jan. 2000.

(3) 博士論文、修士論文

1．鹿野　信太郎：「非構造格子計算法による超音速旅客機形状に関する研究」，
東北大学大学院工学研究科航空宇宙工学専攻博士論文、１９９８年６月。

2．小寺　正敏：「超音速エンジン内部流れ問題への非構造格子計算法の適用に関する研究」，
東北大学大学院工学研究科航空宇宙工学専攻博士論文、１９９９年３月。

3．森野　裕行：「超音速粘性流に対する非構造空間前進解法」，
東北大学大学院工学研究科航空宇宙工学専攻修士論文、１９９９年３月。

4．五十嵐　康隆：「スクラムジェットエンジン内流動数値解析の検証」，
東北大学大学院工学研究科航空宇宙工学専攻修士論文、１９９９年３月。

5．伊藤　靖：「GUI を用いた表面格子生成法」，
東北大学大学院工学研究科航空宇宙工学専攻修士論文、２０００年３月。

6．村山　光宏：「非構造解適合格子法によるデルタ翼高迎角流れの数値シミュレーション」，
東北大学大学院工学研究科航空宇宙工学専攻修士論文、１９９９年３月。

研究成果

１９９７年から１９９９年までの３年間にわたって、標題のような研究をおこなった。その成果を以下に報告する。

研究成果

目次

第１章　序論

1-1 数値流体力学の現状

気象や航空宇宙分野で主体的に始まった数値流体力学（ＣＦＤ）は，特に近年の進歩には目を見張るものがあり，更なる高度化と様々な分野への応用とが加速している．既に航空機や宇宙機，ガスタービン等の開発には不可欠なツールであり，今日では一般の流体機械開発，土木建築といった分野で工学的解析・設計ツールとして認知されるようになったといえよう．また気象，流体物理研究といった流体力学問題に関するあらゆる分野の研究手段として重要な道具になりつつある．コンピュータハードの高性能化とも相まってＣＦＤソフトの工学的ツール化，流体研究ツール化の傾向は今後ますます加速されるであろう．

流体運動を支配する偏微分方程式を差分法なり有限要素法で数値計算を行う場合，解析空間を多数の格子点で離散化して偏微分方程式を代数方程式に変換し，その代数方程式を計算機で解くことになる．数値流体力学のアルゴリズム研究では，ナビエ・ストークス方程式あるいはその近似方程式をどのようにして解くか，また複雑な物理・化学現象をどのようにモデル化するかという議論が中心となる．

ナビエ・ストークス方程式あるいはその近似方程式を計算機で安定に精度良く解く手法の研究開発については近年の進展は著しい．計算のアプローチとしては，航空宇宙関連で用いられる圧縮性流体に対する解法と，低速で密度変化を無視した非圧縮流体解法とに大きく分けられる．前者は保存量（密度，運動量，全エネルギー）を未知数とした連続の式，運動量の式，エネルギー式を解く．一方，非圧縮近似の式は運動方程式と圧力場の式を別に解いて効率的な解法を実現している．

しかしながら，ＣＦＤの適用が依然として特定の流体機械に限られているのも事実である．これは主に複雑な流路形状への対応の困難さから起因している．　翼あるいは翼列などの単純な計算領域・流路形状の場合は，現状のＣＦＤはほぼ実用レベルに達している．しかし，少しでも流路形態が複雑になると作業量が極端に増え，設計現場での使用に耐えられるレベルではない．例えば，ガスタービンでは翼列の三次元多段解析，燃焼器，ミキサーノズル等，既存の構造格子に基づいたＣＦＤでの解析・設計が行われているが，それらはデモンストレーション的な域を越えていない．自動車用エンジン内の流れのシミュレーションも多大なマンパワーを必要としている．今後の見通しとしても，従来の手法を踏襲している限りにおいては作業環境の飛躍的な改善は非常に困難であることも自明である．CFD の工学的ツール化においてもっとも大きなボトルネックとなっているのが格子生成である．

空間を離散化する方法としては，大きく分けて格子点が規則正しく並んだ構造格子（structured grid, 図 1.1(a)）と，並びに規則性を要求しない非構造格子（unstructured grid, 図 1.1(b)）とに分けられる．前者は差分法および有限体積法で用いられ，流体計算では一般的な格子である．後者は有限要素格子とも呼ばれるが，最近では有限要素法以外に有限体積法的なアプローチでも使用されることから，構造格子に対比する意味で非構造格子と一般的には呼ばれている．構造格子では，計算領域を単純な直交格子で離散化する方法もしばしば

採用され，直交格子法（図 1.1(c)）とも呼ばれる．しかし一般的には計算領域の境界と格子線が一致しているものが用いられ，境界適合曲線座標格子（boundary-fitted curvilinear coordinate grid）あるいは BFC 格子とも呼ばれる．

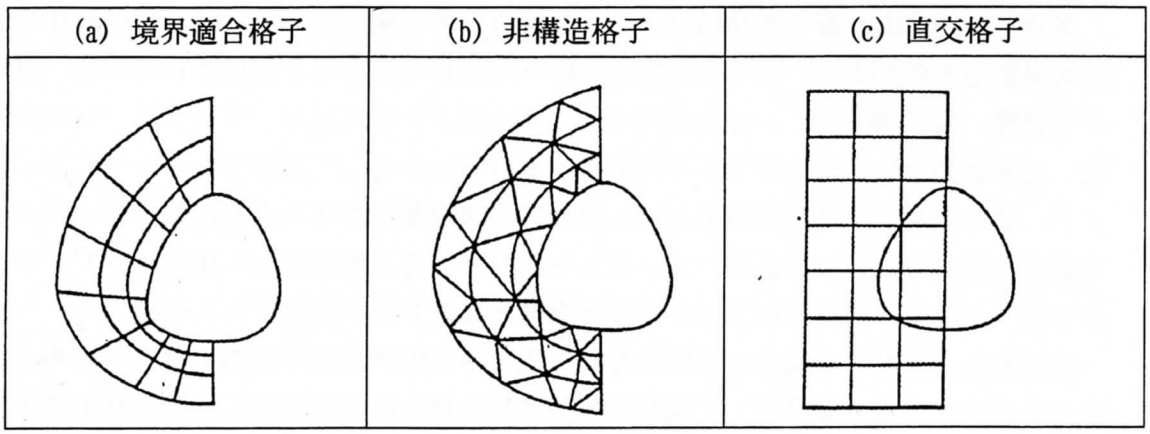


図 1.1 格子の種類

流体計算では流体運動を支配する方程式の強い非線形性のためにその計算時間に多大な時間を要する．計算機の演算能力，記憶容量及び計算アルゴリズムの効率化といった観点から，これまでのところ CFD では構造格子が主流となっていたが，航空機全機周りのような形状に対して構造格子を生成するには，一般的に重合格子(overset grid または chimera grid)法[1-3]やマルチブロック(multi-block)法[4,5]が用いられてきた．しかもこれらのアプローチにはかなりの労力と経験が要求される．

一方，非構造格子は形状融通性に大変優れており，複雑な三次元形状の格子生成も構造格子と比べると遥かに容易である．また非構造格子のもう一つの重要な利点は，任意の領域で格子点の追加及び削除が比較的容易にできることである．そのため流れ場に応じて局所的に格子密度を変化させる解適合格子(solution adaptive grid)法の適用が容易である．

非構造格子法の事実上の研究は 1980 年代半ばから始まり，特に近年，航空宇宙分野を中心として非構造格子 CFD の研究は著しく進展してきている．これは，三次元実形状の高精度な数値計算には格子の幾何学的融通性，格子生成の自動化，および局所的な解適合細分化の容易さという利点から非構造格子が選ばれ，かつこの２０年程に急速に開発されてきた高次精度風上計算法や陰的時間積分法等が非構造格子に拡張されてきたことによる．

1-2 本研究の目的

本研究の目的は，複雑形状への対応法として最も可能性の高い非構造ＣＦＤを設計現場においても使用に耐え得るものまで洗練させること，さらには非構造ＣＦＤと遺伝子アルゴリズム等による最適設計法との組み合わせを通じて，非構造ＣＦＤが真に流体機械設計の手段として成り立ちうるかどうかを試験することである．

現在，非構造格子によるＣＦＤの開発が進んでいる．これは従来の一般曲線座標格子ＣＦ

Ｄの実際問題に対する限界が大きな動機であろう．１９９５年に開催されたCFDに関する国際学会等では，非構造CFDの計算効率の改善や高レイノルズ数流れへの適用，また実作業におけるＣＡＤとＣＦＤとの受け渡しという議論がなされており，非構造ＣＦＤの実用化に向けた研究が着実に進んでいる．本研究の申請者も，非構造格子上での流動計算アルゴリズム，非構造格子形成法，ナビエ・ストークス計算のための非構造格子，表面格子形成法等に関し独自の方法を提案してきた．

これらの研究成果を統合することにより，設計現場でも使用しうる流動解析・設計システムの構築が可能であろうと考えるが，それを実証試験することが本研究の主目的である．従来の構造ＣＦＤと非構造ＣＦＤのいずれが実用に耐えうるかの議論は最近よく問われる点であり，二者択一的な回答は余り意味がないが，ＣＦＤ実用化に対する実証例を本試験研究において出すことは，今後のＣＦＤ研究の方向性および流体機械設計手段の進展を左右する大きなキーポイントとなろう．

1-3 研究の進め方

本研究は，非構造ＣＦＤによる流体機械の流動解析・設計システムを構築し，それを現場レベルの設計作業に適用して非構造ＣＦＤの有用性を試験することを目的としており，研究は以下の３つの作業に大きく分けて進めた．

(1) 非構造CFDによる流動解析・設計システムの構築，
(2) 非構造ＣＦＤと最適化法の組み合わせ，
(3) 流体機械設計への応用による非構造CFDの有用性評価試験．

非構造ＣＦＤによる流動解析・設計システムは，更に格子生成とソルバー開発，改良とに分けられる．格子生成では，既開発の空間非構造格子生成コード[6]の信頼性を大幅に改善する．また，格子生成の実作業では表面格子生成，およびその前段階としてのモデリングがもっとも手間と時間を要する．この状況を改善するため，既開発の表面格子生成法[7]に基づき，ＣＡＤ用ワークステーションの出力から直接に表面格子生成を行う方法を開発する．これにより，CADから格子形成および流動解析までの一貫した解析システムを構築する．また，ナビエ・ストークス計算用のハイブリッド格子生成法を開発し，高レイノルズ数流れの非構造解析を可能にする．

流体機械の最適化は工学的に大きな期待が寄せられている．三次元実形状に対して，以下に形状最適化を導入するかを検討，開発する．

また，これら開発コードを元に，様々な流体機械，流体要素等の実際の問題に適用し，これら解析・設計過程での問題点を明らかにし，非構造ＣＦＤシステムの有用性を総合的に論ずる．

1-4 本報告書の構成

本研究課題の研究においていくつかの世界的に誇れる成果が得られている。しかし、それ

らを総て記述するにはかなりのページ数を必要とするため、ここでは本研究課題において開発された根幹要素についてのみ詳述し、その検証・試験、応用・展開研究についてはダイジェスト的に記述する。つまり非構造 CFD の根幹をなす格子生成を第２章において、またソルバーを第３章で詳述し、第４章では、それら非構造 CFD の応用展開研究結果を巻末に添付した代表的発表論文の解説で代用する。第５章はまとめである。

参考文献

[1] Benek, J. L., Steger, J. L. and Dougherty, F. C., "A Flexible Grid Embedding Technique with Application to the Euler Equations," AIAA Paper 83-1944, 1983.

[2] Benek, A., Buning, P. G., and Steger, J. L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85-1523, 1985.

[3] Cao, H. V., Su, T. Y., and Rogers, S. E., "Navier-Stokes Analyses of a 747 High Lift Configuration," AIAA Paper 98-2623, 1998.

[4] Lee, K. D., "3-D Transonic Flow Computations Using Grid Systems with Block Structure," AIAA Paper 81-0988, 1981.

[5] Sawada, K. and Takanashi, S., "A Numerical Investigation on Wing/Nacelle Interferences of USB Configuration," AIAA 87-0455, 1987.

[6] Sharov, D. and Nakahashi, K., "A Boundary Recovery Algorithm for Delaunay Tetrahedral Meshing," 5th International Conference on Numerical Grid Generation in Computational Field Simulations, 1996, pp.229-238.

[7] Nakahashi, K., and Sharov, D., "Direct Surface Triangulation Using the Advancing Front Method," AIAA Paper 95-1686-CP, 1995, pp.442-451.

第２章　非構造格子生成

2-1 非構造格子ＣＦＤの前処理

数値流体力学（Computational Fluid Dynamics; CFD）は今日，計算機の大幅な進歩とともに様々な形状まわりの流れ場に対して適用され，その設計や解析に欠かせないものとなってきている．この躍進の理由として，風洞実験や飛行実験に比較してより少ない費用で流れ場の解析が行えること，流れを乱すことなく様々な物理量の情報が得られること，計算結果にある程度の信頼性があることが挙げられる．だが実際の設計環境において，CFD の有効性が最大限に発揮されることは稀である．この最大の原因として，計算機援用設計（Computer Aided Design; CAD）システムと格子生成ツール，数値計算ツール，後処理ツールが一つの有用な設計環境として十分に統合されていないことが挙げられる．図 2.1 に CFD 計算過程を示す．特に，CAD と格子生成ツールの密接な連帯が格子の精度と作業時間の観点から非常に重要になる．

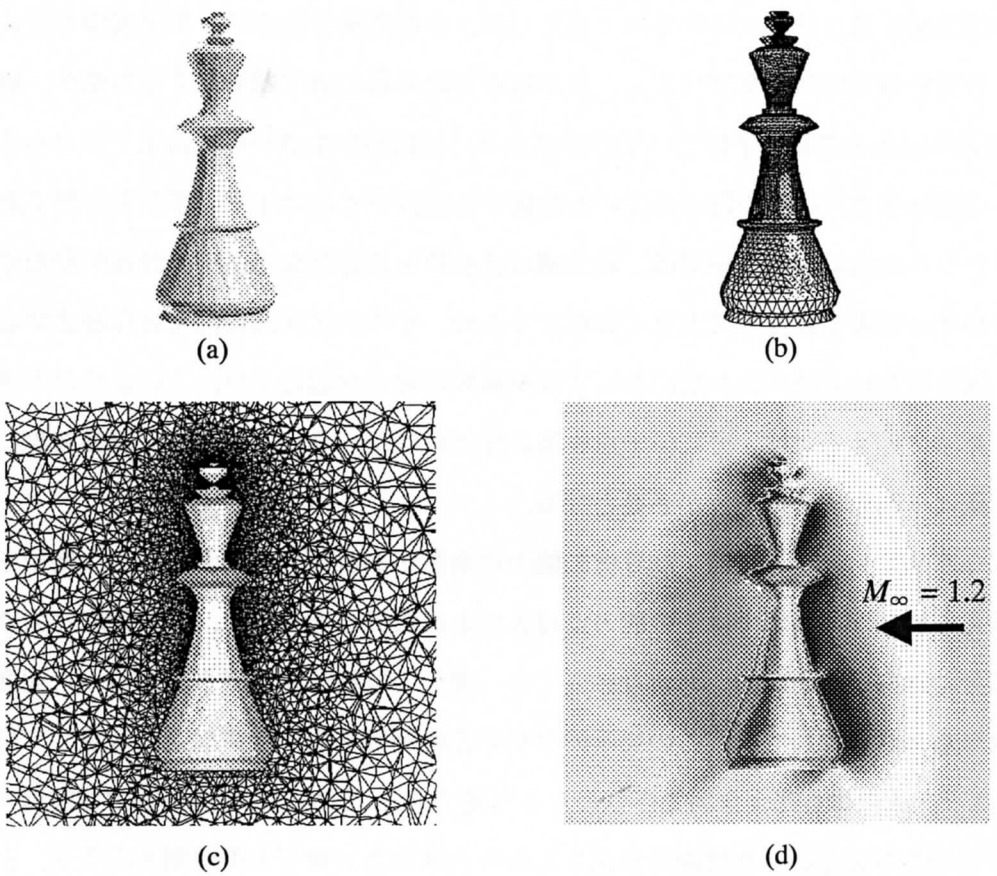


図 2.1　チェスの駒を例にした CFD 計算過程：(a) 表面形状定義，(b) 表面格子生成，(c) 空間格子生成，(d) 計算と後処理

さらに計算前処理として行わなければならない格子生成は，三次元複雑形状に対しての難しさと，格子密度制御の問題などで多大な手間が要求されることから，CFD の工学的ツール化の大きな障害となっている．現状では実際，格子生成に要する時間は，流れ場を Euler や NS などの方程式を用いて計算するよりも多くなる傾向にある．

一方，面倒な格子生成の必要がない，空間を単に x, y, z 座標方向に切る直交格子法が計算機の発展に伴って最近再び注目されている．しかし航空機形状に適用する場合には，計算精度の観点から表面の滑らかさを保つことが非常に重要となる．また，壁付近などの物理量変化の激しい部分で格子の細分化または境界適合格子とのハイブリッド化などをなるべく行わずに，直交格子法のアルゴリズムの大きな長所である簡易さを損なわないことも重要であると考えられる．このように，直交格子法は現時点の計算機の性能では航空機形状に適用した場合，克服すべき問題点がまだ数多く存在する．

2-2 非構造格子法での表面格子生成

このことから伝統的な境界適合格子法に替わって，近年では三次元複雑形状に対してより適応度が高い非構造格子法が盛んに研究されてきている．非構造格子法では，形状表面を三角形セルで分割し，計算領域を四面体やプリズム，またはその組み合わせで離散化する方法が一般的である．現在，四面体要素の空間格子についてはほぼ完全に自動的な生成が可能なまでになっている．

しかし，空間格子生成の前段階としての表面格子生成はいまだに手間の要する作業である．特に航空機まわりの流れの数値解析では，航空機の性能算出に必要な表面圧力分布や壁面摩擦係数などが表面格子の密度や質などに大きく依存するため，その生成作業には十分な注意が必要となる．また，遷音速や超音速の流れ場においては衝撃波の発生が起こるが，このような不連続面を離散的な計算法で捉えるには十分に格子点を集めなければならない．同様に，境界層流れや自由剪断層の解像にも高密度の格子点が必要となる．

このような問題に対処するには，計算機の能力に余裕があれば一様に十分細かな格子を作成することも考えられるが，現状ではその能力にはまだまだ限界があるので，計算精度に大きく影響を与える部分には格子点を集め，あまり必要でない部分では格子点を減らすといった配慮が必要となる．簡単な形状に関しては，この局所的な格子密度制御を面の曲率などを考慮して自動的に行い，表面格子生成自体も完全自動で行うことも可能である．しかし航空機形状では前縁や後縁，また胴体と翼の接合部といった稜線を無視して面の曲率のみで格子密度制御を行うと，表面格子で表現される形状面が稜線部分で凸凹になってしまうという問題点がある．論文[1]では曲率を考慮した表面格子自動生成を行っており，再突入機への適用例があるが，前縁の稜線を考慮してい

ないため前縁の部分で凸凹が見られる．加えて，完全自動でユーザーの思い通りに格子点密度を制御することにも限界がある．このように格子密度制御を考慮した表面格子生成作業は一般的に面倒であり，表面格子生成に手間がかかる大きな原因の一つである．

また，翼形状の定義などにも高い精度が要求されるが，設計現場で多用されている CAD データが必ずしも表面格子生成に活かされてはいない．表面格子生成を簡素化する意味でも，CAD データを効率的に表面格子生成過程で用いる工夫が必要である．

本研究では，非構造表面格子を CAD データから効率的かつ効果的に作成する手法を開発することを目的とする．具体的には，以下の四点に重点を置く．

- ユーザーの負担を最小限に押さえる
- 計算精度に大きく影響する格子密度制御を容易に行う
- 格子の質を上げる
- 格子生成時間の短縮を図る

2-3 CAD 出力形式

表面格子生成を効率的かつ効果的に行うには，CAD で作成した三次元形状を手際よく利用する必要がある．CAD 出力には様々な形式があるが，その中から最も一般的に使用されている IGES（Initial Graphics Exchange Specification）形式と，形状を比較的簡単に表現することができる STL（STereoLithography）形式について考えてみたい．

(1) IGES 形式

IGES 形式は，CAD/CAM 間で図形データなどをやり取りするための中間ファイルとして用いられる．IGES の扱う要素は，幾何要素・表記・構造の三つに分けられる[5].

- 幾何要素（20 種）

  円弧，複合曲線，円錐曲線，有意点列，平面，線分，スプライン曲線，スプライン曲面，点，線織面，回転面，柱面，変換行列，線形経路，閉領域，フラッシュ，有理 B スプライン曲線，有理 B スプライン曲面，節点，有限要素点
- 表記（13 種）

  角度寸法，中心線，直径寸法，フラッグ注記，引き出し注記，注記，引き出し線，長

さ寸法，基準寸法，点寸法，半径寸法，断面表示，寸法補助線

- 構造（10 種）

  結合の定義と具体形，図面，線フォント，マクロの定義と具体形，特性値，子図の定義と具体形，方状配列，環状配列，文字フォント，投影図

このIGES形式を用いた表面格子生成法がいくつか提案されている(例として, [1], [6]-[8]). IGESデータは形状の詳細な幾何情報を保持できるが，現在の IGES 規格ではソリッドモデリングやパラメータ情報などをサポートしていないため，表面格子生成に利用する場合には適切な表現方法に変換する必要がある.

(2) STL 形式

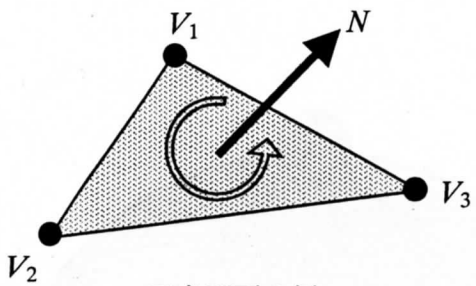
STL 形式は CAD から造形装置にデータを渡す場合に用いられる書式であり, ラピッドプロトタイピングにおける事実上の標準形式である[9]. STL ファイルは，ASCII 形式でもバイナリー形式でも出力可能であり，三次元自由曲面を三角形面の集合体として近似する. STL データの構成を図 2.2 に, また簡単のために近似した三角形面が 2 つだけの場合の STL データ例を図 2.3 に示す. これらの図で示されているように STL ファイルは，それぞれの三角形セルに対してセルの単位法線ベクトルと，そのベクトルが外向きを表す方向から見て反時計回りに 3 つの節点座標を与え，純粋に形状に関する情報のみから成り立っている. そのため，表面格子生成の際に利用することが比較的容易にできる.

CAD で作成された形状を STL データに変換する際に，CAD 内部表現の形状と三角形面近似形状の幾何学的誤差がユーザーによって指定された値ξに収まるように，各々の三角形面の大きさと形が決められる. そのため，十分な精度でもとの形状を近似することができるが，STL データで用いられる三角形面を制限する要素はこのξのみである. したがって図 2.4 示したように，例えば平面は非常に粗い三角形で表現され, また円柱面や円錐面は細長い三角形の集合で表現される. そのため，STL データではすでに形状表面が三角形分割されているとはいっても，直接 CFD 計算用の表面格子として用いることができることはまずあり得ない. ここでは，STL データを CFD 表面格子生成のための背景格子として用いることとした.

さらに STL を利用するもう一つの利点として，形状表面を三角形面の集合体として表すという特性から，構造表面格子やパネル計算法に用いる表面格子のような形状データを STL 形式のデータに変換して利用することも非常に簡単であることが挙げられる. このことも STL データを入力書式とする利点といえよう.

```
SOLID <solid name>
    <List of Facets>
ENDSOLID
```

(a) STL フォーマット（ASCII）



(b) 三角形面の例

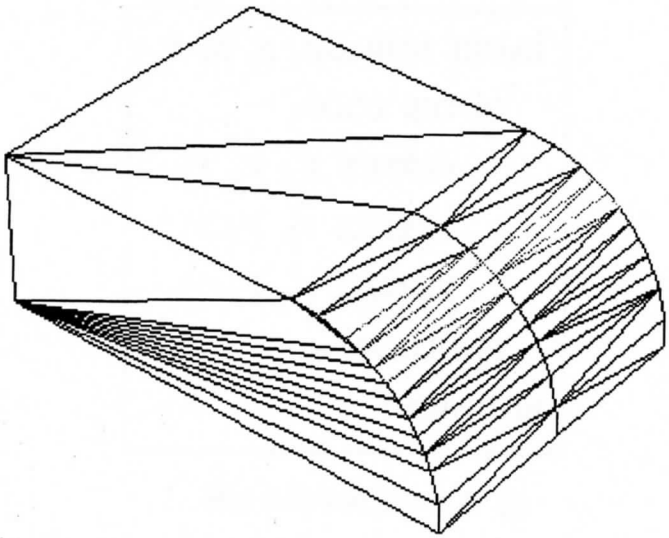
```
FACET NORMAL Nx Ny Nz
   OUTER LOOP
      VERTEX V1x V1y V1z
      VERTEX V2x V2y V2z
      VERTEX V3x V3y V3z
   ENDLOOP
ENDFACET
```

(c) 各々の三角形面の情報

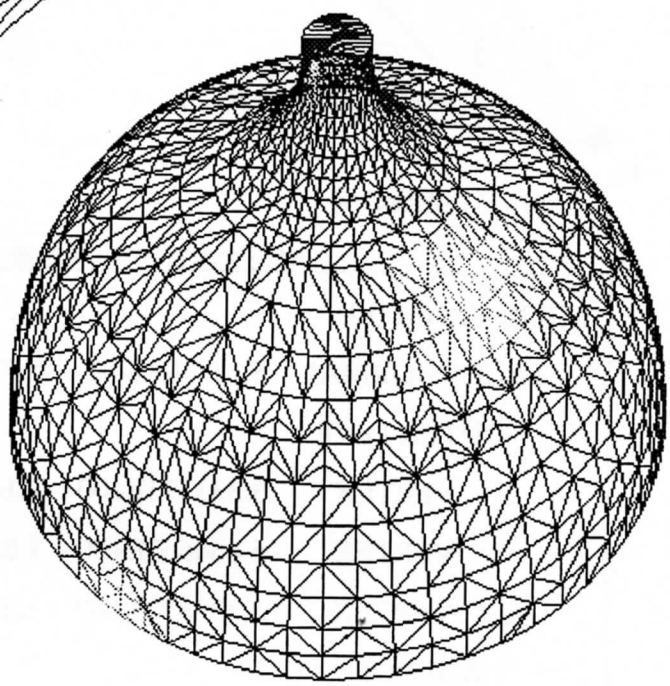
図 2.2 STL データの構成

```
SOLID EXAMPLE
   FACET NORMAL  5.773503E-01   5.773503E-01   5.773503E-01
      OUTER LOOP
         VERTEX     3.333333E+00   7.146973E-12   6.666667E+00
         VERTEX     6.666667E+00   1.053291E-11   3.333333E+00
         VERTEX     3.333333E+00   1.666667E+00   5.000000E+00
      ENDLOOP
   ENDFACET
   FACET NORMAL  5.773503E-01   5.773503E-01   5.773503E-01
      OUTER LOOP
         VERTEX     6.666667E+00   1.053291E-11   3.333333E+00
         VERTEX     6.666667E+00   3.333333E+00   2.440270E-13
         VERTEX     3.333333E+00   1.666667E+00   5.000000E+00
      ENDLOOP
   ENDFACET
ENDSOLID
```

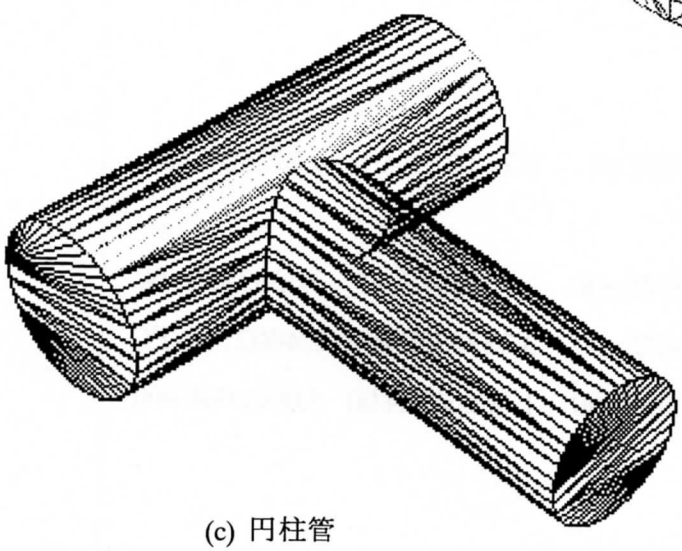
図 2.3 STL データの例



(a) 箱



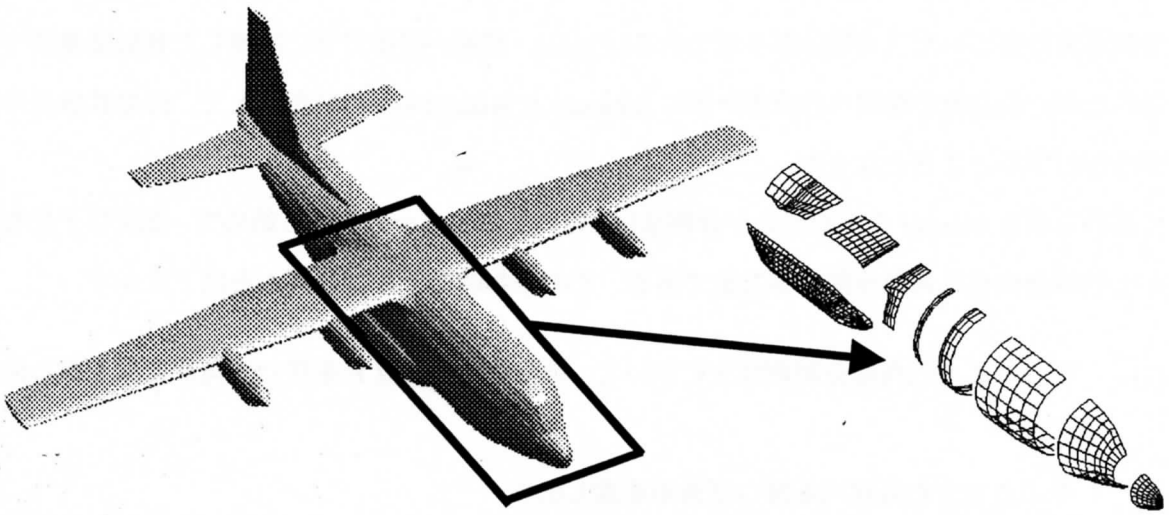
(b) 瓶



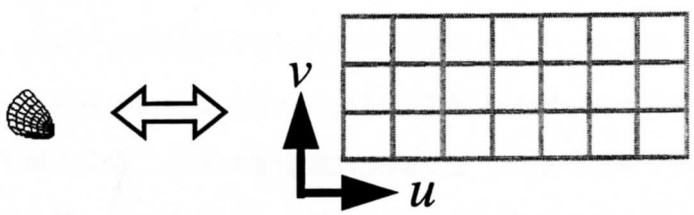
(c) 円柱管

図 2.4 STL データで表現される形状の例

2-4 従来の表面格子生成法



(a)



(b)

図 2.5　写像法での格子作成手順の模式図：(a) 三次元形状面をパッチに分割－胴体全部のパッチのみ表示してある，(b) 各々のパッチを二次平面に写像

任意の三次元表面に三角形非構造格子を生成する方法は様々提案されているが，そのもっとも基本的な手法は，三次元表面を適当な領域に分け，それぞれの曲面を２次元平面に写像する方法（mapping method）である．この写像法の概要を以下に示す．

(1)　三次元対象表面を多数のパッチと呼ばれる幾何学的に矩形な領域に分割する（図 2.5a 参照）

(2)　それぞれのパッチを二次平面に写像する（図 2.5b 参照）

(3)　その二次平面上で三角形分割を行う

(4)　生成した格子をもとの三次元表面に再写像する

二次元平面上の三角形分割は，デローニー三角形分割法か前進先端法のいずれかが用いられる．

デローニー法（Delaunay triangulation）は，“空間に任意におかれた格子点群に対し，それらを結んでできあがった三角形の外接円内に他のいずれの格子点も含まないような分割ができる”という数学的事実に基づいた三角形分割手法である[12][13]．最初，固体力学に関連した有限要素法で開発されたが，航空機全機周りの流体計算に Jameson と Baker[14]が用いたことで，他の数値流体研究者の注目するところとなった．

よく用いられる Bowyer のデローニー分割法は，粗いデローニー分割から始めて，逐次格子点を追加して領域全体を三角形分割する方法である．その基本的なアルゴリズムを以下に示す．

(1) デローニー三角形分割領域上において，新たに加える点 P を含む三角形要素を探し出す

(2) その点を含む外接円を持つ三角形を探し出す

(3) 探し出された三角形要素から共有辺を取り除いて多角形を作り，格子点 P とその各頂点を結んだ三角形を作る

以上の作業を繰り返すことによって，領域全体で細かな三角形分割ができる．

ただし，この方法は完全に凸な単一領域での三角形分割の場合にのみ成り立つ．一般の問題では，境界に凹な部分を持つ場合が多い．また例えば翼型まわりに二次元で格子生成を行う場合，翼面の内部境界と外部境界の間の領域を考える．このような場合には，翼面境界を横切って三角形分割を行わないような制約を課さなければならない．また曲面に対してデローニー法を適用する場合，デローニー条件を満足する三角形を生成できるとは限らない．そのため任意の三次元形状面に対するデローニー分割はできない．

このようにデローニー法にはきれいな三角形で分割が行われるという利点があるが，表面格子生成に利用しようとした場合，解決しなければならない問題も多い．

一方，前進先端法（advancing front method）は，ある閉じた空間において要素形成の先端（front）を境界から領域内部へと順次前進させていく方法で，すべての領域が要素で埋め尽くされるまで前進を続ける．基本的な方法は以下の通りである．

図 2.6 において，境界において格子点の分布が与えられ，それぞれに格子番号が付けられているとする（図 2.6a）．また隣同士を結ぶ辺はフロントと呼ばれ，フロントの集まりは領域を完全に囲んでいる．格子形成の過程では，このフロントは格子形成の前面にあってその辺を底辺とする三角形を作り得るアクティブな辺のことを指し，従って順次領域内を前進していく．各フロントは前進方向に対して左側の格子点番号，続いて右側の格子点番号を記憶している．

次に，フロント辺から一つの辺を選び，それを底辺とした三角形を作る（図 2.6a の点線）．通常

はもっとも短い辺を選ぶ．この新しい頂点 C の近くに他の頂点がない場合はこの点を採用し，フロントリストを更新する．つまり，フロントリストから(4-5)を消去し，新しいフロント(4-15), (15-5)をリストに加える．

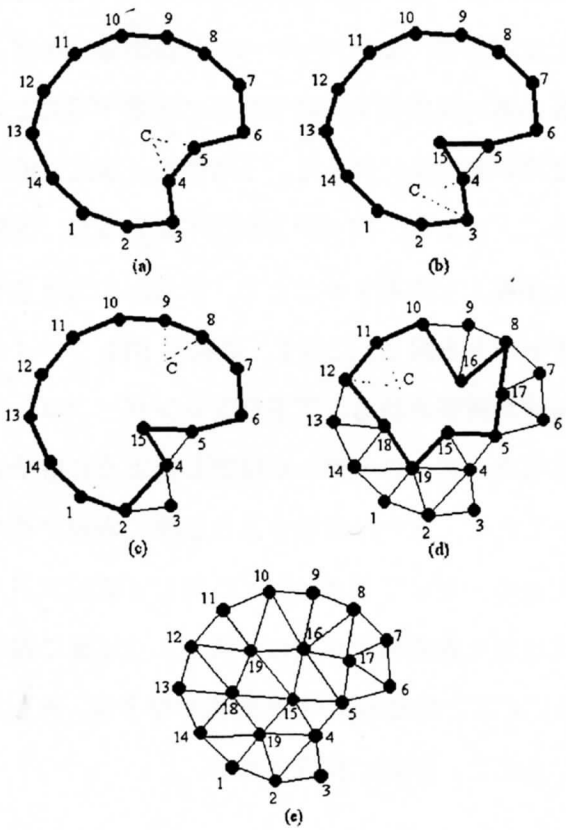


図 2.6 前進先端法による格子生成

次に，同様に新しい三角形候補 3-4-C（図 2.6b）を考える．この場合，C の周りに頂点 2， 15，あるいは 1 といった頂点があり，これらのいずれかと底辺 3，4 を結んでも三角形を構成できる．これらのうちのどれを選ぶかは後で述べる．図 9.3.11c の例では，頂点 2 が選ばれて三角形を作っている．以上の手続きをフロントリストが空になるまで繰り返す．

前進先端法は，境界から順次格子を形成していくことから，境界近くでの格子形状の制御が比較的やさしい．そのため壁境界が重要である流体計算に向いていると言えよう．しかしながら，この方法では局所的な最適化のもとに形成されているため，必ずしもきれいな格子を作り得るとは限らない．形成後の平滑化等の格子後処理が必要な場合もある．また，近接点を捜したり要素辺の交わりをチェックする等の演算のために，デローニー法の格子形成法に比べて時間がかかるのが欠点である．また，格子生成が行き詰まることがしばしば起こる．その場合は，生成手順を

ある程度戻すなり，局所的に細かな格子を作るなりの手段が用いられる．

2.5 幾何形状特徴線を利用した表面格子生成法

写像をもちいた方法では，二次平面上で三角形分割を行うため，非常にアルゴリズムが簡単になるという大きな利点がある．しかし，CFD 計算結果に対して重要となる局所的な格子密度制御が必ずしも容易に行えるとは限らず，またパッチへの分割作業は非常に面倒である．さらにこのパッチ分割は，最終的に質の高い表面格子を得るために必要不可欠というわけではない．

ここでは，従来の写像を用いる方法に代えて，三角形を三次元の曲面に直接張っていく方法を開発した．写像を用いないことにより，パッチ分割は不要となり，大幅な作業時間の短縮が期待できる．また，直接三次元曲面上で作業することで，三角形の大きさ等を制御が容易となろう．

表面格子生成のフローチャートを図 2.7 に示す．最初に STL ファイルなどの形状データを読み込む．そのデータから，稜線が構築される．アドバンシング・フロント法を適用する際に必要となる初期フロントを作成するために，ユーザーが指定したように格子点がこれらの稜線上に配置される．格子密度を制御するため，または形状をより正確に表現するためにソースラインやソースポイントの追加も可能である．そして三次元形状に対して直接アドバンシング・フロント法が適用される．その後，作成された表面格子が元の形状をより正確に表現するために，形状復元が適用される．空間格子生成に必要となる外部･対称面境界格子は，あらかじめ用意されているテンプレートから形状を選択したあと，同様に作成される．

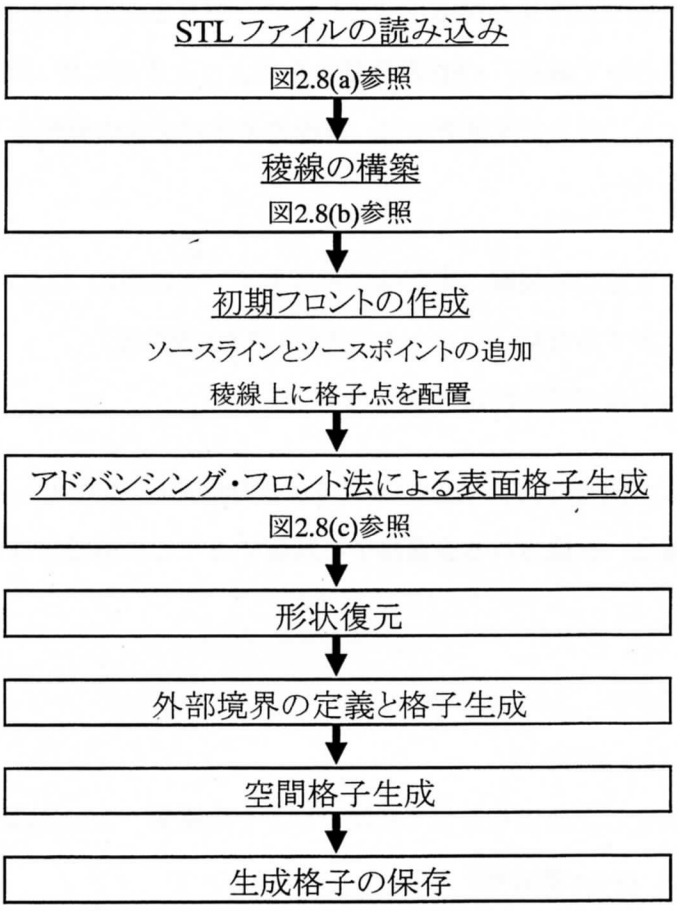


図 2.7　表面格子生成のフローチャート

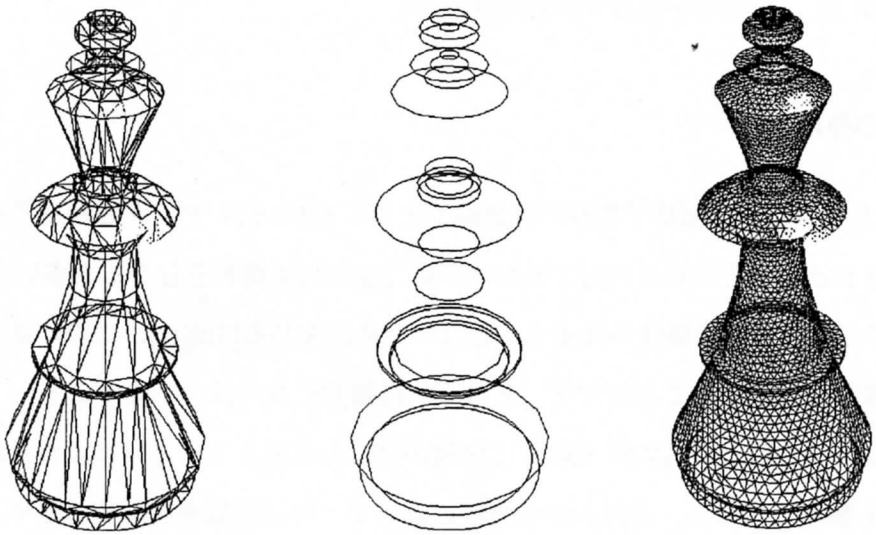


図 2.8　チェスの駒に対する表面格子生成例：(a) 典型的な STL データー粗い三角形や非常に細長い三角形，潰れた三角形を含む　(b) 稜線の構築　(c) 作成した表面格子

格子生成は，CFD 計算プロセスを簡略化するために，できるだけ自動的に行う必要がある．しかし限られた計算資源の中で最高の CFD 計算結果を得ようとするには，局所的な格子密度制御が非常に重要となってくる．そこで本研究では，表面格子生成の次の過程をマウスやキーボードを用いた会話型処理で行う．

- ソースラインすなわち稜線，またはソースポイントの追加（2.5.3 節参照）
- 各々の稜線に対する分割パラメータの指定（2.5.3 節参照）
- 最大または最小格子長の指定（2.5.4 節参照）

これらの処理に必要な GUI 環境は，Visual C++® 6.0 と OpenGL™ を使用して構築した．

次に示す残りの過程は，作成される表面格子を制御することに直接関与しないので，自動的に行う．

- STL データの修正（2.5.1 節参照）
- 稜線の構築（2.5.2 節参照）
- ユーザーが指定した値にもとづく初期フロントの構築（2.5.3 節参照）
- 表面格子生成（2.5.4 節参照）
- 形状復元（2.5.5 節参照）

以下の節において，表面格子生成の詳細を述べる．

2.5.1 背景格子の作成

2.3 節で述べたように，表面格子生成の背景格子として CAD-STL データを利用するが，STL データには背景格子としてそのまま利用できない不都合な三角形面を含むことが多い．これは，STL ファイル形式では整合性や完備性の検査を行わないために生じる[10][11]．この好ましくない三角形は，二つの種類に分類することができ，それぞれ自動的に取り除く必要がある．

第一の種類は，一つ以上の内角が 1°以下で面積がほとんど零となっているような非常に細長い三角形（図 2.9 参照）である．これらの三角形によって，後に稜線を構築する際や，表面格子生成の段階で新たな格子点位置を求める際に，適切な計算が行えない可能性がある．非常に細長い三角形は，次のように検出し，修正する．

(1) すべての辺の長さを計算する

(2) 各々の三角形面において，長さについて降冪に三つの辺を並べる

(3)　次の参照値ξを用いて，細長い三角形を検索する

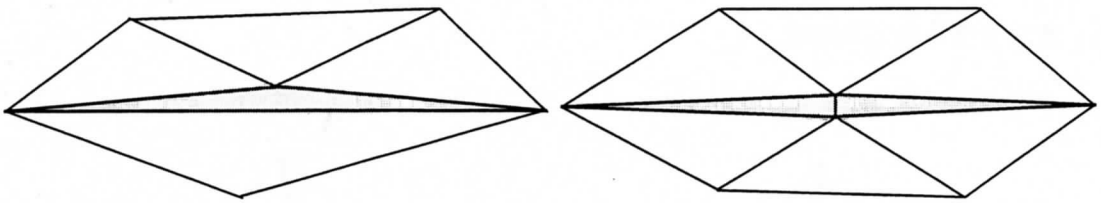
$$\xi = \frac{l_2 + l_3}{l_1 \cos\alpha}$$

ここで，l_1, l_2, l_3（$l_1 > l_2 > l_3$）は，その三角形面が持つ三つの辺の長さであり，αは1.5úのような小さな角度を与える．もしξが1よりも小さいとき，その三角形は非常に細長いとみなす．

(4)　もし$l_3 > 0.05\, l_1$のとき，図 2.9a に示されているような孤立した非常に細長い三角形とみなす．この三角形は，図 2.10a のように一番長い辺を交換することで修正される．

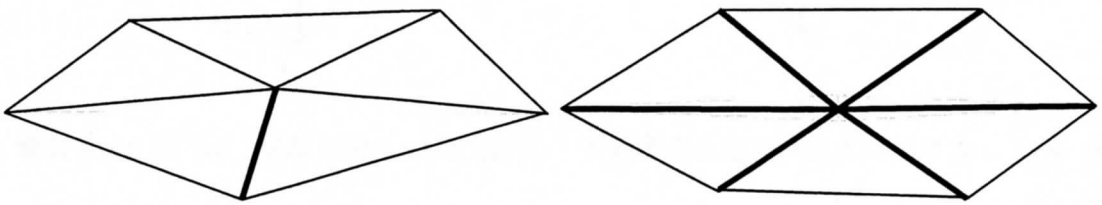
(5)　もしl_3　$0.05\, l_1$のとき，図 2.9b に示されているような隣にもう一つ同じような三角形が並んだ非常に細長い三角形とみなす．これらの三角形は，図 2.10b のように二つとも取り除く．

このような非常に細長い三角形の面積は，先に述べたようにほとんど零であるので，これらの修正によって形状の正確さが損なわれることはない．



(a)　(b)

図 2.9 STL データの中に含まれる非常に細長い三角形面のモデル:(a) 孤立した非常に細長い三角形面，(b) 二つが並んだ非常に細長い三角形面

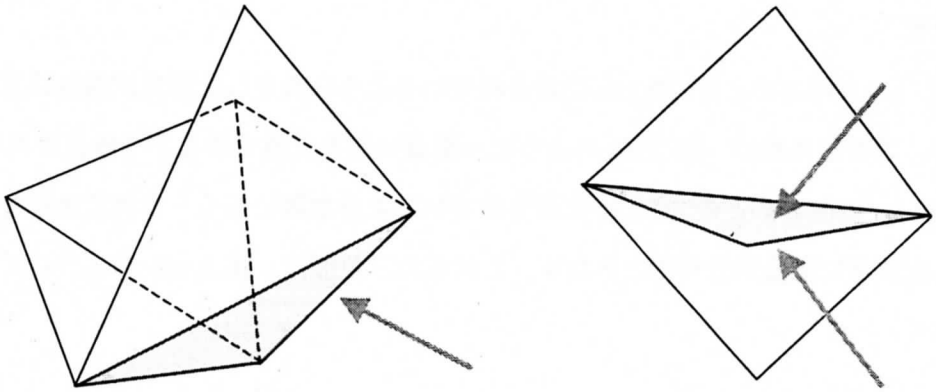


(a)　(b)

図 2.10　非常に細長い三角形面の自動修正：(a) 図 2.9a の修正結果，(b) 図 2.9b の修正結果

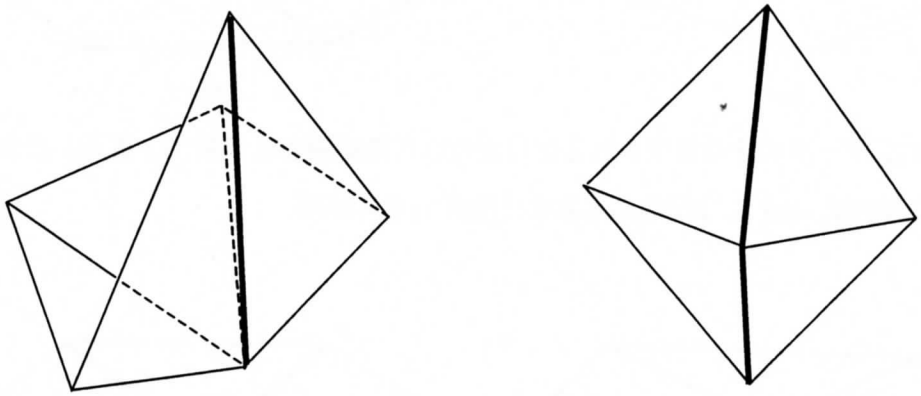
第二の種類は，張り付いた三角形（図 2.11）である．これらの三角形は，それぞれの法線ベクトルを用いて検索される．まず，すべての隣り合った二つの三角形面の法線ベクトルがなす角を

計算し，160ú以上の場合にはその二つの三角形に印を付ける．そして，もし三つの印を付けた三角形面が三つの節点を共有している場合，そのうちの一つは飛び出した三角形面（図 2.11a）であり，図 2.12a のように一番長い辺を隣の三角形面と交換することで修正される．また，もし二つの印を付けた三角形面が三つの節点を共有している場合，それらは重なっており（図 2.11b），図 2.12b のように修正される．



(a) (b)

図 2.11 STL データの中に含まれる張り付いた三角形面の例：(a) 飛び出した三角形面，(b) 重なり合った三角形面



(a) (b)

図 2.12 張り付いた三角形面の自動修正：(a) 図 2.11a の修正結果，(b) 図 2.11b の修正結果

これまでに述べてきた不都合な三角形面の自動修正をする前に，STL データからは以前述べたように，それぞれの三角形面の単位法線ベクトルと 3 つの節点座標の情報が得られるだけであるので，効率よく処理を行うために，総節点数，辺情報（両端の節点番号と両側のセル番号），セル情報（まわりを囲む 3 つのセル番号と辺番号），および節点情報（曲率と単位法線ベクトル）を作

成する必要がある．

図 2.13 に，本研究で作成したアプリケーションの全体像を示す．この図の中でウインドウに表示されている機体は，NAL で設計された小型超音速実験機モデル[21]の STL データである．STL 形式の出力は，最近のほとんどの CAD ソフトでサポートされているが，ここではフランスのダッソー社によって開発された CATIA を用いた．

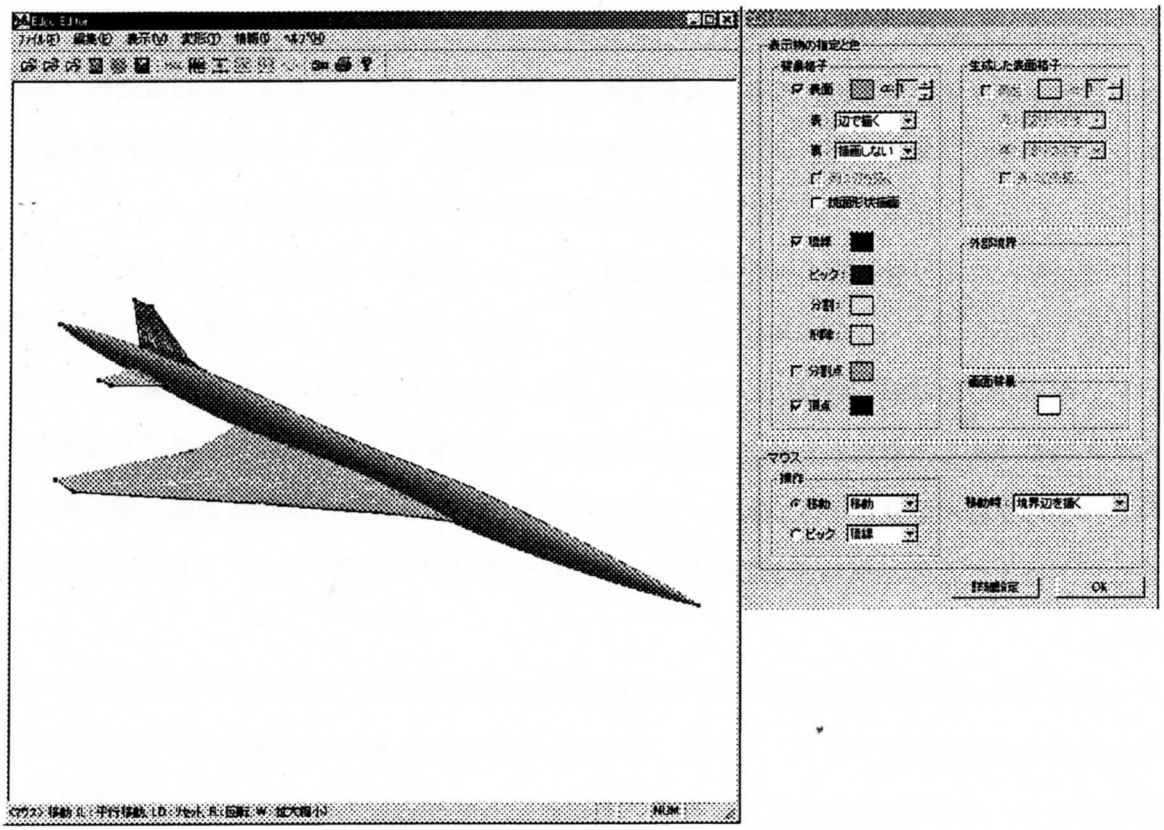


図 2.13 ウインドウに STL データを表示した表面格子生成アプリケーションの全体図

2.5.2 稜線の作成

第二ステップでは，STL データから図 2.14c のような稜線データを作成する．稜線は形状表面の対称面での境界線や，翼前縁または後縁，翼と胴体の結合線といった形状面の折れ曲がり部を指す．この稜線は，次節において表面格子生成に必要となる初期フロントを形成するために，格子点を配置する基準として利用される．そのため元の物体形状を表面格子においても正確に表現するために正確な構築が必要となる．例えば，飛行機の翼の後縁や翼と胴体の接合線等は，生成される表面格子でも角として扱われなければならない．また翼の前縁も，前縁線を通さずに表面格子を生成すると，流れ場の計算において前縁近傍の圧力分布等の精度が著しく損なわれること

が多い．

稜線は，境界辺と呼ばれる辺を連結したものとして定義される．境界辺には実境界辺と内部境界辺とがある．実境界辺は，その辺の片側にしか三角形セルが存在しないような辺である．内部境界辺は，基本的には表面の折れ曲がった部分を指し，その取り出しは，辺を挟む三角形セルの単位法線ベクトルの内積がある閾値以下のものとする．このようにして定義した初期稜線の例を図 2.14a に示す．この図において稜線上に付けられた四角い点は，稜線の境界点か，あるいは稜線の折り曲がり点を示す．

初期稜線を構成する境界辺のうち，内部境界辺は閾値の設定により大きく変わるため，画面上で生成稜線の確認と閾値の変更という会話処理が必要となる．また，一般的には稜線候補辺が多めに挙げられることが多く，以下の判定条件で不必要な内部境界辺を取り除く．

まずそれぞれの境界候補辺の中点間距離を調べる．この距離が，選んだ 2 つの候補辺の短い方の長さ以下の場合，どちらかの候補辺を取り除く．どちらを取り除くかの基準は，候補辺両端の節点につながっている他の候補辺の本数が少ないか，候補辺を挟むセルの単位法線ベクトルの内積が大きいかである．次にもし候補辺同士が節点を介して繋がっている場合，両辺のなす角度を調べる．この角度が非常に小さい場合，先に述べた基準でどちらかの候補辺を取り除くが，翼端部分などで本当に必要な候補辺を削除しないように気を付ける必要がある．この結果が図 2.14b である．

この段階で残った境界辺は，孤立していたり途切れ途切れになっていたりする場合が多いため，さらに必要な境界辺を追加する．これは，特に丸い前縁をもつ翼での前縁線の取り出しに不可欠な作業である．この追加作業はまず，稜線の端にあたる境界辺と節点を探す．この境界辺の単位方向ベクトルを端の節点を基準にして $\bar{\mathbf{s}}$ とする．次にその端の節点に繋がっている辺を探す．この中のある辺の端の節点を基準にした単位方向ベクトルを $\bar{\mathbf{s}}_i$，両側のセルの単位法線ベクトルをそれぞれ $\bar{\mathbf{n}}_{i1}$，$\bar{\mathbf{n}}_{i2}$ として，次の参照値 ξ_i を計算する．

$$\xi_i = 0.5\left(\bar{\mathbf{s}} \cdot \bar{\mathbf{s}}_i\right) + \bar{\mathbf{n}}_{i1} \cdot \bar{\mathbf{n}}_{i2}$$

この参照値が最小の辺を境界辺にする．この稜線の追加作業を止めるのは，他の稜線に出会ったときや，$\cos^{-1}\left(\bar{\mathbf{n}}_{i1} \cdot \bar{\mathbf{n}}_{i2}\right)$ の値が指定された角度（初期値では 3û）以下になったときである．

最後にそれぞれの境界辺をつなぎ合わせて稜線とする．また次節において，稜線上に格子点を分布させるときにスプライン補間を用いるので，稜線の折り曲がり点も（図の四角点）も取り出しておく．図 2.14c は最終的な稜線を示す．以上の作業は，実際には利用者がメニューのボタンを押すだけで自動的に行われる．

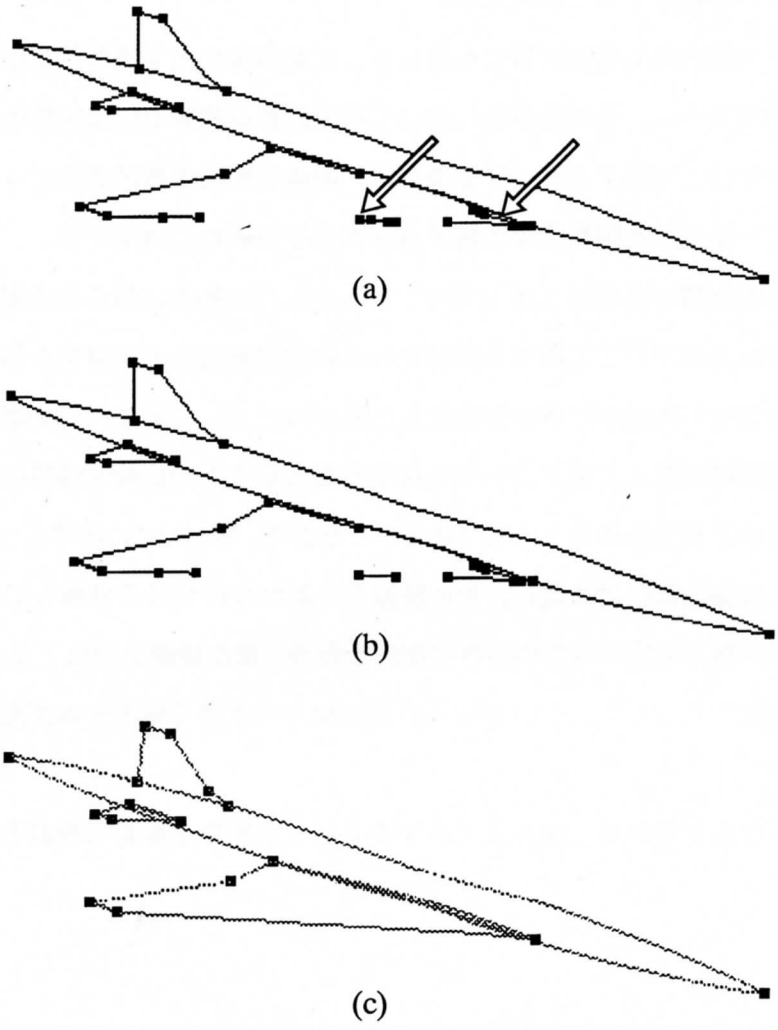


図 2.14 NAL超音速実験機モデルにおける稜線生成過程：(a) 初期稜線－稜線が途切れて，また不必要な稜線も存在する，(b) 不必要な稜線の削除後，(c) 最終的な稜線

2.5.3 初期フロントの生成

第三ステップでは，アドバンシング・フロント法を適用する際に必要となる初期フロントを構築する．前の過程で得られたそれぞれの稜線を画面上でマウスを用いてピック（選択）して，その稜線上に格子点を分布するか，あるいはその自動的に得られた稜線が不必要であるならば削除するかを指定する．格子点を分布させる場合は，格子点数と稜線両端における格子幅の指定を行う．そしてスプライン補間を用いて稜線を一次元に変換し，Vinokur の一次元補間関数[22]により格子点を分布させたあと，元の曲線上に戻す．Vinokur の一次元補間関数は，線分上に指定された格子点を，線分両端での指定された格子幅を保持し，かつその間の格子幅は滑らかになるような分布を与える．稜線上のこれらの格子点は，アドバンシング・フロント法による表面格子生成の

ための初期フロントを構成することになる．

稜線は表面格子の局所的な密度制御にも用いる．主翼前縁などで局所的に格子密度を上げる場合は，境界辺の追加を行い，その境界辺によって形成される稜線上に指定数の格子点を分布させる．また，稜線の中ほどで格子を密にする場合は，稜線を形成する節点をピックすることで稜線を二分割し，上記一次元補間関数による格子点分布をそれぞれに適用する．

これらの格子密度制御の例を図 2.15 に示す．図 2.15a は自動的に得られる稜線であるが，二次元の正方形領域であるので，この領域を囲むまわりの実境界辺のみが稜線を形成する．この稜線上に格子点を分布させて内部面に格子を生成したのが図 2.15b である．この場合，稜線に近い部分での格子分布制御は稜線上の格子点分布の様子を変えることで比較的容易に行えるが，正方形領域の中心部分に格子点を集めることはできない．そこで，図 2.15c に示すように，背景格子の任意の辺を選択して稜線を新たに作成し，その稜線上に細かな格子点を分布して図 2.15d に示すような中心部分に高い格子密度領域を作る等の制御を行う．また稜線上の格子点の分布も，図 2.15e のように稜線の分割することで，図 2.15f のような稜線上で大きく格子分布が変わる格子が得られる．

以上述べたような入力操作は，GUI 上でのマウスピック操作が重要な役割を担っている．

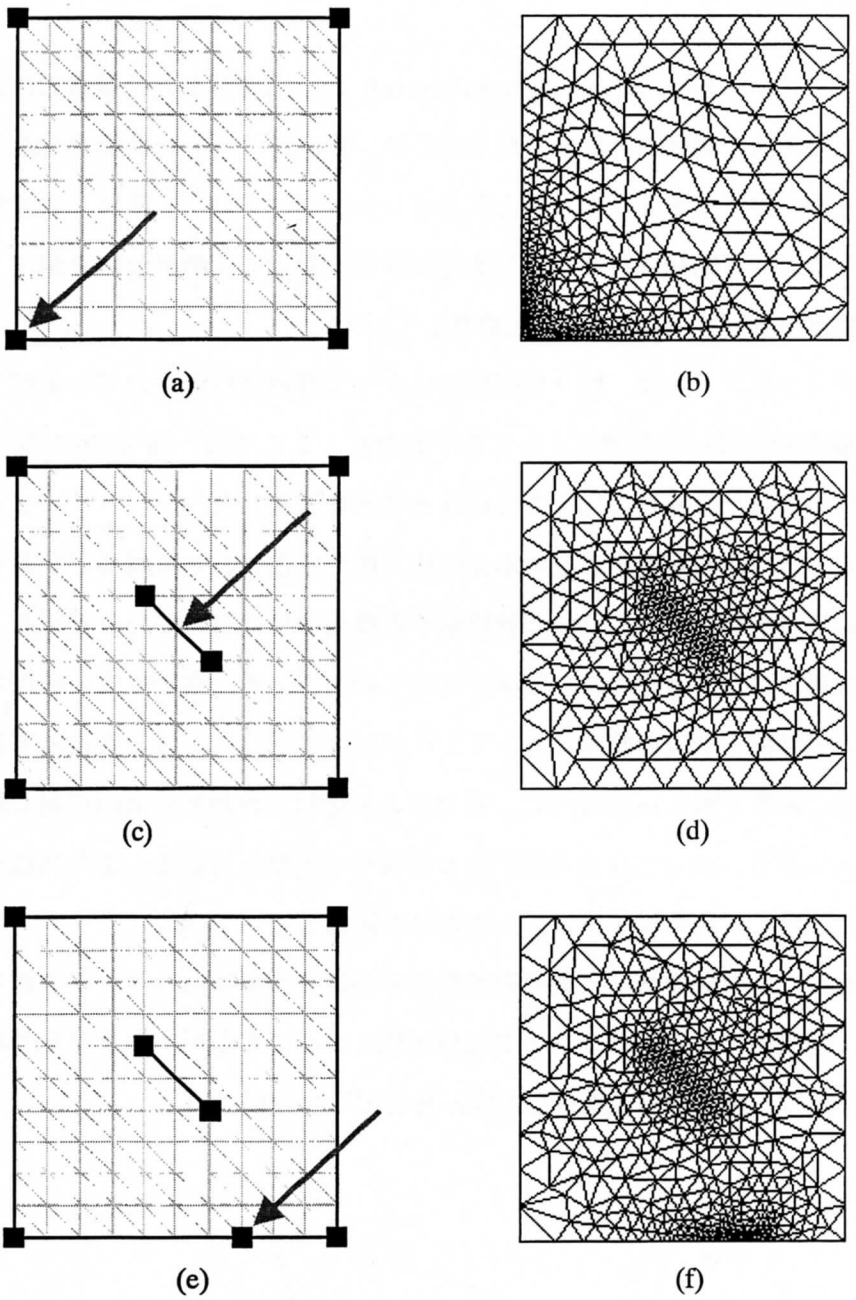


図 2.15 ソースラインまたはソースポイントの追加による局所的な格子密度制御例：(a) 正方形領域－太い実線で表されるようにすでに稜線が自動的に検出されている，(b) a を元にした格子生成例－稜線近くでの格子密度制御は非常に容易に行える，(c) ソースラインの追加－自動検出された稜線から遠い中心部でも格子密度制御を容易にする，(d) c を元にした格子生成例，(e) ソースポイントの追加－稜線上に急激に格子点を集めたいときに有効である，(f) e を元にした格子生成例

2.5.4 三角形分割

アドバンシング・フロント法（advancing front method）は，三角形などの要素の作成を初期フロントと呼ばれる要素生成の開始点の集まりからはじめ，逐次フロントを前進させながら要素で領域を埋め尽くしていく方法である．もともとアドバンシング・フロント法は二次元領域に三角形格子を生成する方法として開発されたが，三次元形状面に対して三角形分割を施すことや，三次元空間を四面体要素で分割する[16]ことにも成功している．

アドバンシング・フロント法は，数々の研究者によって詳細に報告されているが（例として，[17]-[20]），その細部の手順は研究者によって若干異なる．ここでは，論文[20]で提案された方法をもとに，いくつかの新たな改良を加えて信頼性と効率を改善した．

いま，格子生成を行う面を定義する三角形面の集合体で定義された背景格子があり，その境界において格子点の分布が与えられ，それぞれに格子点番号が付けられているとする．格子点を結ぶ辺の集まりは初期フロントと呼ばれ，領域を完全に囲んでいる．格子生成の過程において，これらフロント上の辺は格子形成の前線にあって，それぞれのフロント辺を底辺とする新たな三角形を作り得る状態にある．図 2.16 のように，各フロント辺は三角形生成方向に対し面の表から見て左側の格子点 A の番号，続いて右側の格子点 B の番号を記憶している．また同様に左側の隣接フロント辺番号に続いて右側の隣接フロント辺番号を記憶している．

格子を形成する前に，対象領域に三角形要素形状を制御する形状パラメータを背景格子の節点上に分布させる．現段階ではこのパラメータは格子幅 δ のみであるが，格子点を効率よく配置するためには，引き延ばし度 s や引き延ばし方向 α を指定した方がよい．

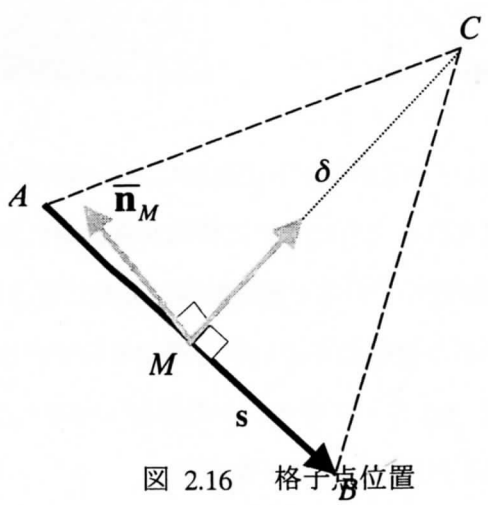


図 2.16　格子点位置

新しい三角形は以下の手順で形成される．

(1)　すべてのフロント辺の中からいちばん短いものを選び，その両端の節点をそれぞれ A,

B とすると，AB の中点 M での格子幅 δ_M を背景格子から内挿して決める．

(2) AB を底辺として高さ δ の二等辺三角形を作り，その頂点を C とする．δ はその位置での定義面などの曲率を考慮する必要があるが，基本的に以下の基準で決められる．

$$\delta \leftarrow \begin{cases} 0.55AB & (\delta_M < 0.55AB) \\ \delta_M & (0.55AB \le \delta_M < 2AB) \\ 2AB & (2AB \le \delta_M) \end{cases}$$

また C の位置は，AB の単位方向ベクトルを $\bar{\mathbf{s}}$ とし，中点 M を含む背景格子上の三角形セルの単位法線ベクトルを $\bar{\mathbf{n}}_M$ とすると，次のように求められる．

$$\mathbf{r}_c = \mathbf{r}_M + \delta(\bar{\mathbf{n}}_M \times \bar{\mathbf{s}})$$

ここで，定義面上に写像した節点 C がきちんとその面上に存在することを確認する．曲率が非常に大きな面などでは，写像した節点位置が一意に定まらない場合があるので，δ を逐次小さくして必ず節点 C が定義面上の一点に存在するようにする．

(3) MC を内分する点 Q を以下のように求める．ただし，l_A, l_B はそれぞれ節点 A, B で隣接しているフロント辺の長さである．

$$\mathbf{r}_Q = \gamma \mathbf{r}_C + (1.0 - \gamma)\mathbf{r}_M$$

$$\gamma = \begin{cases} 0.5 & (\gamma' < 0.5) \\ \gamma' & (0.5 \le \gamma' < 1.0) \\ 1.0 & (1.0 \le \gamma') \end{cases}$$

$$\gamma' = \begin{cases} l_A/\delta & (l_A < l_B) \\ l_B/\delta & (l_A \ge l_B) \end{cases}$$

そして点 Q を中心として，AB の長さを $|\mathbf{s}|$ としたときに半径 nr

$$r = \begin{cases} 10.0|\mathbf{s}| & (10.0|\mathbf{s}| < 0.7\delta) \\ 0.7\delta & (10.0|\mathbf{s}| \ge 0.7\delta) \end{cases}$$

$$n \sim 15.0$$

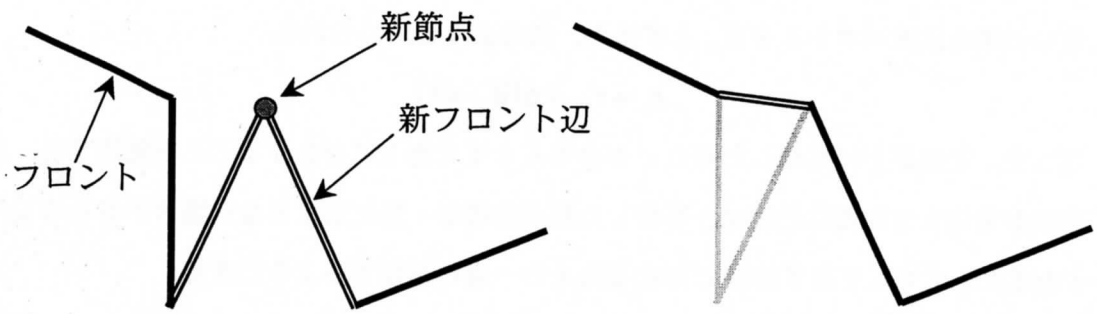
の円内にあるすでに作成された格子点を探索し，その結果得られた p 個の節点を点 Q から距離が近い順に $N_1, N_2, \cdots, N_p$ と順に並べ，最後に点 C を加え近接点リストを作成する．

(4) 近接点リストから，三角形 ABN_i（$i = 1, \cdots, p+1$）を考える．これらの三角形の中で次の条件を満たす最初の節点 N_i を選び，新しい三角形を形成する．

- 三角形 ABN_i 内部に，節点 C を除いた他の節点 N_j（$j = 1, \cdots, p$）を含まない
- 辺 AN_i, BN_i が既存のフロント辺と交差しない
- 三角形 ABN_i はなるべく細長い三角形でない

(5)　新たに作成されたフロント辺がある場合，それぞれの隣接フロント辺の状態を確認する．図 2.17 に示されるように，後に作成される三角形がどうしても潰れてしまう場合がある．このようなフロント辺をそのまま残しておくと，表面格子生成が失敗する原因となることがある．そこで隣接フロント辺とのなす角が 20úよりも小さい場合，すぐに新たな三角形を作ってしまうようにする．

(6)　フロントリストを更新する．



(a) 新しい節点の作成　　(b) 潰れた三角形の作成

図 2.17　新節点の作成により生じる潰れた三角形の形成

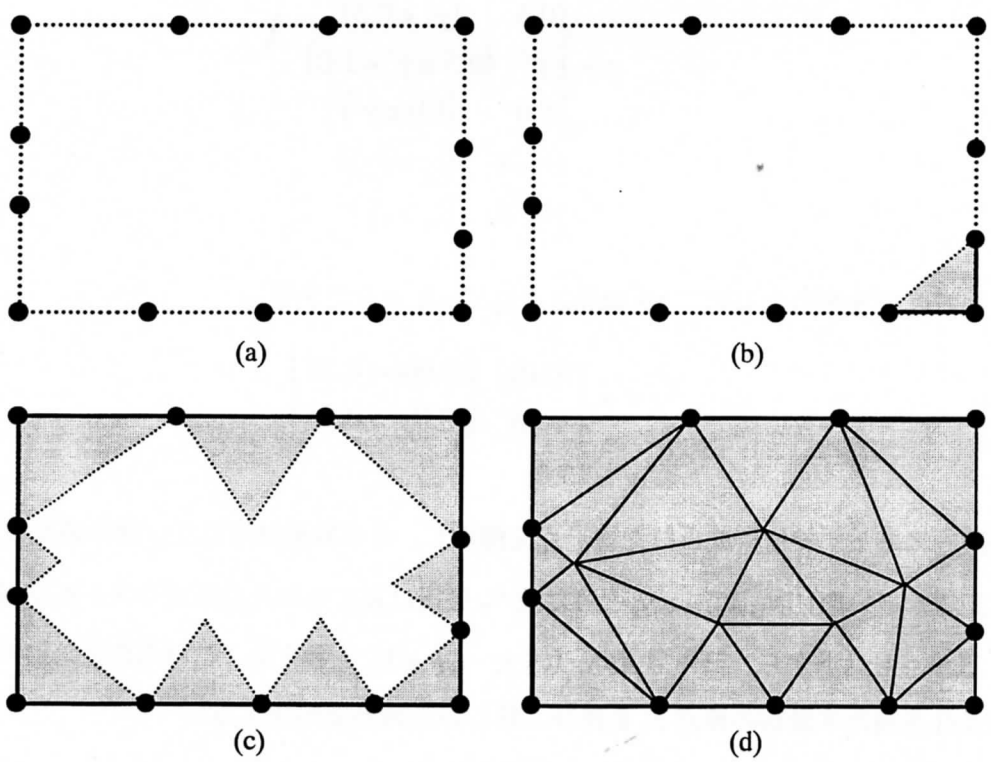


図 2.18　アドバンシング・フロント法の適用例：(a) 長方形領域の初期フロントー黒丸は配置した節点を，破線はフロント辺を表す，(b) 最初の三角形の形成ー初期フロントの中でいちばん短い辺をもとにする，(c) 初期フロントがなくなった段階，(d) 最終的な格子

以上のステップをまず初期フロントに対して適用し，初期フロント辺がすべてなくなった後に，残りのフロント辺がすべてなくなるまで適用する．こうして，領域全体に三角形格子が生成される．格子生成の簡単な例を図 2.18 に示す．

アドバンシング・フロント法では，境界から順次格子を生成していくことから，境界近くでの格子形状の制御が比較的容易に行える．そのため，壁境界が重要となる CFD 計算に向いていると言えるであろう．

本研究では，このアドバンシング・フロント法を表面格子生成の際のアルゴリズムとして採用する．その利点を改めてまとめると，以下の通りである．

- 三次元形状をパッチに分割する必要がない
- 境界近くでの局所的な格子密度制御が比較的容易に行える
- 表面の曲率を考慮した格子生成が可能である

しかし，アドバンシング・フロント法には一般的に以下のような問題点が存在する．

- 近接点リストを作成したり，フロント辺の交差を判定したりする演算に時間を要する
- 格子生成が行き詰まる場合がある
- 局所的な最適化で格子生成を行うため，きれいな格子ができるとは限らない
- 初期フロントの構築に手間がかかる

最初の三つの問題は，プログラムを改善することで実用的には問題にならない程度の対応が可能である．例えば，新しく作られるフロント辺が他のフロント辺と交差しているかどうかの判定を考えてみる．この交差判定をすべての既存フロント辺と行っていたのでは，大規模な格子生成の際にはこの判定に多くの計算時間が必要となる．そこですべての既存フロント辺と比較するのではなく，フロントによって形成される閉領域を逐次考えることにより，それぞれの閉領域を囲むフロントごとに交差判定を行い，生成時間を大幅に短縮することができる．また，面倒な初期フロントの構築に関しては，できる限り自動的に行う必要性がある．

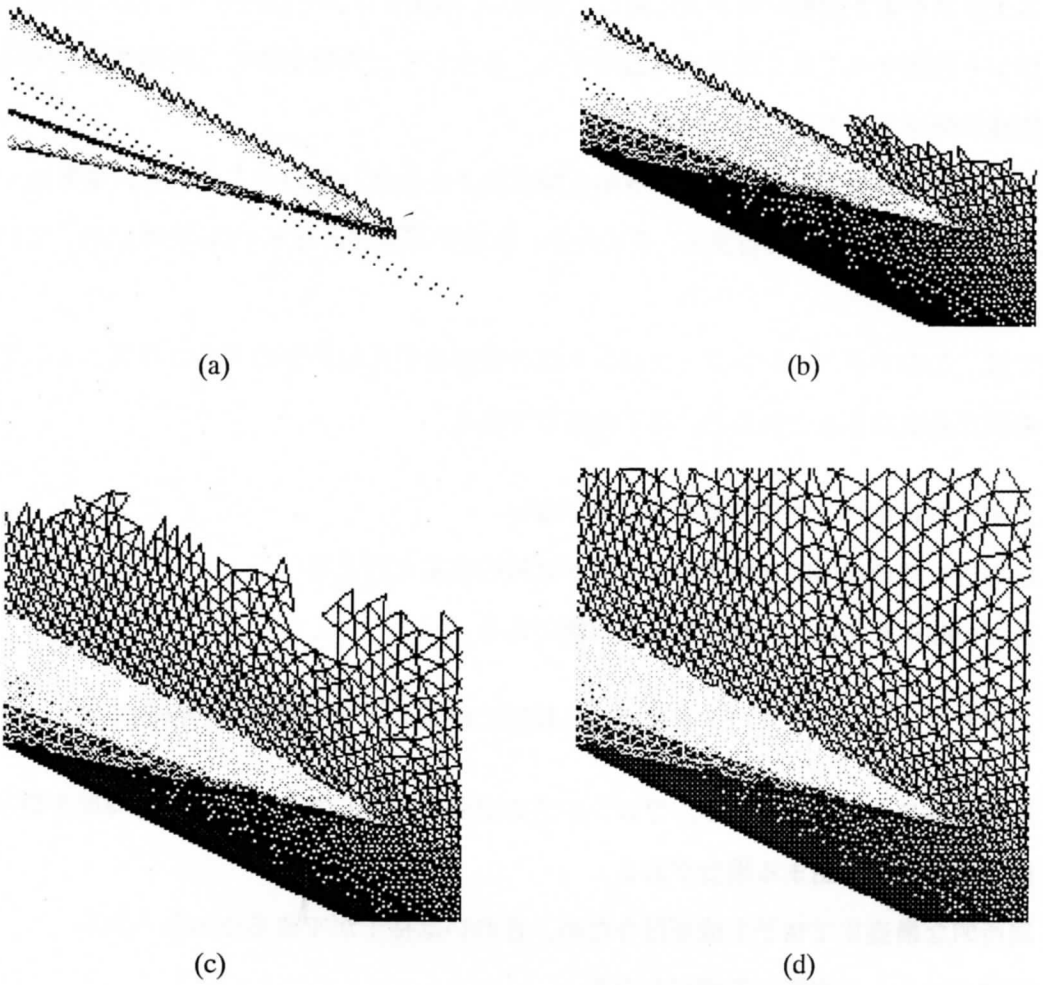


図 2.19 NAL 超音速実験機モデルの前縁部における三角形分割：(a-c) 格子生成過程－フロントが前進している様子が分かる，(d) 格子生成終了

第四ステップでは，第三ステップで構成した初期フロントからアドバンシング・フロント法を用いて実際に表面格子生成を行う（図 2.19 参照）．物体面が稜線で囲まれた閉領域である場合，それぞれの閉領域ごとに格子幅の最大値および最小値を指定して格子密度制御を行うことができる．

2.5.5 形状復元

表面格子生成過程において，新たに作成される格子点位置は本手法の堅牢性保持のために，一時的に背景格子から線形補間を用いて決められる．しかし，このままでは図 2.20a のように背景格子が非常に粗い場合，作成した表面格子にその影響が表れてしまう．そこで，格子生成後に二次補間で格子点位置を決定し，格子精度の改善を図る．この形状復元を行うことによって，背景格

子が粗い場合でも図 2.20c のように非常になめらかな格子を得ることができる．

この形状復元は，次のような計算で行う．復元後の格子点位置は，二次三角形形状関数で与えられる[23]．まず中点位置 $\mathbf{r}_M$ は，ハミルトンの多項式を用いて計算する．

$$\mathbf{r}_M = 0.5\mathbf{r}_1 + 0.125\mathbf{s}_1 + 0.5\mathbf{r}_2 - 0.125\mathbf{s}_2$$

ただし，

$$\mathbf{r}_i = |\mathbf{s}| \frac{\mathbf{n}_i \times (\mathbf{s} \times \mathbf{n}_i)}{|\mathbf{n}_i \times (\mathbf{s} \times \mathbf{n}_i)|}, \ (i = 1, 2), \ \mathbf{s} = \mathbf{x}_2 - \mathbf{x}_1$$

であり，$\mathbf{x}_1$, $\mathbf{x}_2$ は辺の両端の節点位置，$\mathbf{n}_1$, $\mathbf{n}_2$ はそれぞれの節点位置での単位法線ベクトルである（図 2.21 参照）．

そして，ある格子点 i が背景格子のある三角形セル上にあるとき，その面積座標を(h_1, h_2, h_3)，それぞれの辺の中点座標を $\mathbf{r}_{M1}$, $\mathbf{r}_{M2}$, $\mathbf{r}_{M3}$ とすると，形状復元後の格子点 i の座標は次のように表される（図 2.22 参照）．

$$\begin{aligned}\mathbf{r}_i' = {} & h_1(2h_1 - 1)\mathbf{r}_1 + h_2(2h_2 - 1)\mathbf{r}_2 + h_3(2h_3 - 1)\mathbf{r}_3 \\ & + 4h_1h_2\mathbf{r}_{M1} + 4h_2h_3\mathbf{r}_{M2} + 4h_3h_1\mathbf{r}_{M3}\end{aligned}$$

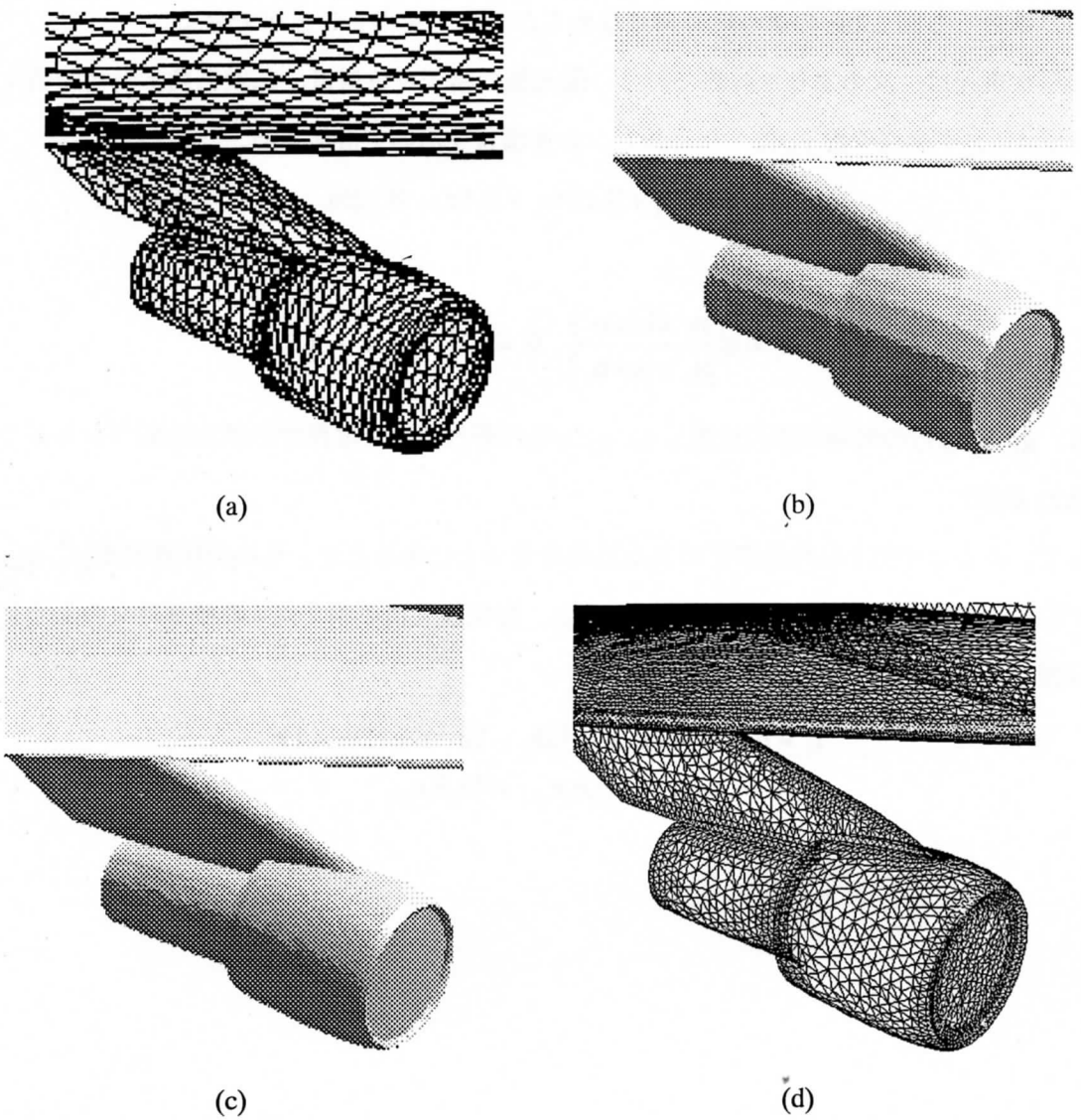


図 2.20　Boeing 747 の内側エンジンナセルに対する表面格子生成：(a) 背景格子－STL データではないため，非常に粗く形状を表現している，(b) 作成された表面格子－粗い背景格子の影響がはっきりと表れている，(c, d) 形状復元後の表面格子－非常になめらかな格子になっている

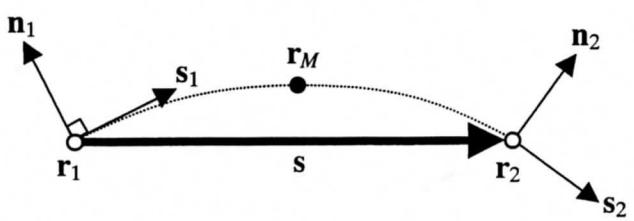


図 2.21　中点位置 $\mathbf{r}_M$

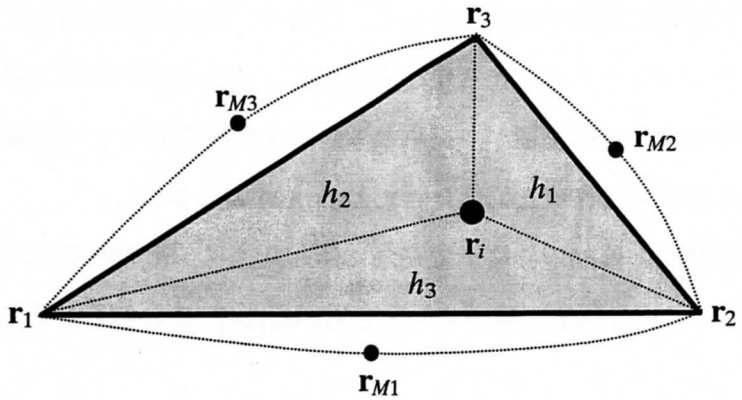


図 2.22　形状復元時の座標関係

こうして，背景格子のそれぞれの三角形セル頂点での法線ベクトルで定義される曲面上に格子点が写像されることになる．しかし，三角形セルのある節点が後縁などのような鋭い稜線上に存在するとき，その中点位置をこの方法で求めると，図 2.23 の破線で表されるように背景格子から大きくはずれてしまう．このようなときには格子点での法線ベクトルとして一番近いセルのものを代わりに採用する．

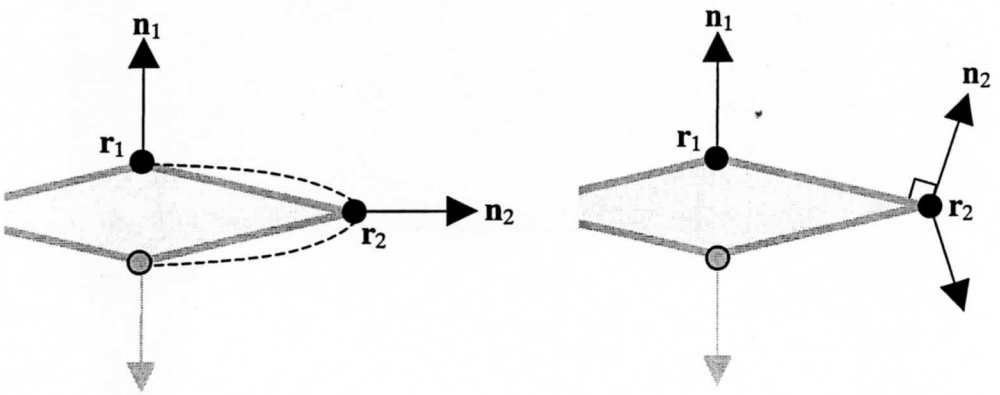


図 2.23　鋭い稜線上での形状復元：(a) 二次補間，(b) 鋭い稜線上にある節点では，法線ベクトルとして一番近いセルのものを採用する

2.5.6 外部境界と空間格子の作成

空間格子生成では，物体面上の表面格子の他に，計算領域を完全に覆うような外部境界表面格子も必要となる．この外部境界の作成を効率良く行うために，外部境界として用いられる基本的

な形状，例えば亜音速・遷音速計算用には半球や全球，超音速計算用には半円錐や全円柱など，をテンプレートとして事前に用意する方法をとる．つまり，例えば対称面を仮定した飛行機周りの遷音速計算では，テンプレートの中から半球を選択し，その中心の座標（初期値として，対称面上における背景格子の中心点の座標が与えられる）と半径を指定して外部境界を定義する．その後の作業は物体面の表面格子作成と同様に行われる（図 2.24 参照）．

計算領域を物体表面格子と外部境界格子で完全に囲んだ後は，その内部に四面体空間格子を生成する．ここではデローニー分割法に基づく手法を用いた[24]．この空間格子生成法については次節で述べる．

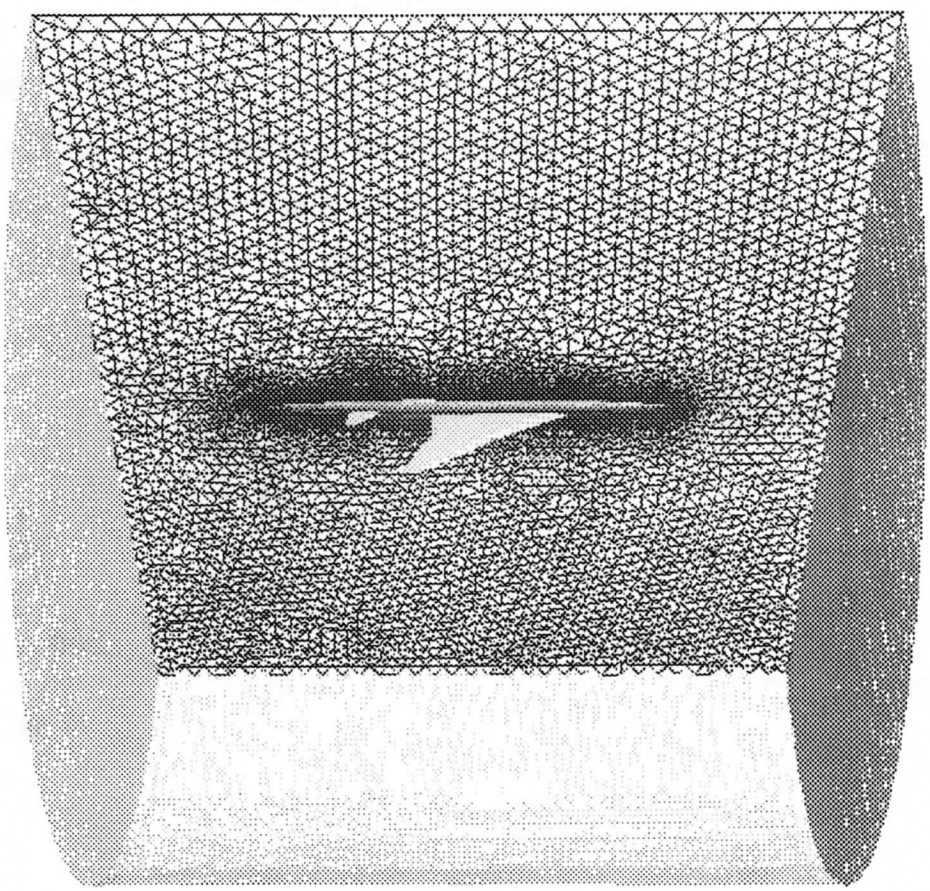


図 2.24 NAL 超音速実験機モデルの表面格子

2.6 空間格子生成

2.6.1 Introduction

Two basic methods are usually used for automatic volume grid generations - advancing front and Delaunay methods. These methods have demonstrated their efficiency for two-dimensional unstructured meshing. Unfortunately their extension to three dimensions is a challenge. Among the most trouble features of the three-dimensional triangulation are the lack of any tetrahedral construction for some surface triangulations, the examples are shown in Fig. 2.25, the lack of volume Delaunay triangulation for the arbitrary surface triangulation, and the lack of efficient tool to eliminate bad shaped tetrahedra which are obtained during the generation process. The first feature may lead to some problem for advancing front methods while the second one causes troubles for Delaunay methods.

The methods based upon the advancing front concept build new elements by growing up or advancing a front which initially represents the boundary surface triangulation. This method has shown its capabilities to create three-dimensional grids which exactly conform the given boundary surface triangulation. However the efficiency of the method is bounded considerably by the use of search algorithms to check the possible intersections between the current front and the newly formed element. Moreover, sometime it is impossible to find an appropriate triangulation for the front (see Fig. 2.25) without insertion of new point. That can produce very distorted tetrahedra or can lead to code failure in some pathological cases.

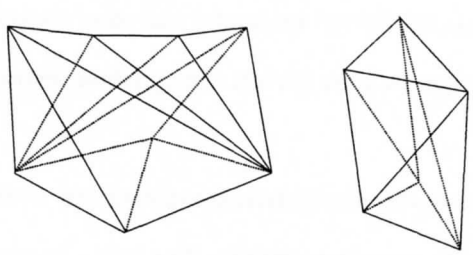


Fig. 2.25 Sample of triangulated surfaces providing no tetrahedra inside

The second approach is based upon the Delaunay concept [25-27]. This technique connects the given points to form a topologically valid non-intersecting set of tetrahedra. This method is much faster as compared to advancing front method because all operations are performed locally without global search algorithms, and generation process for realistic three-dimensional geometry takes about 1 CPU hour on workstation [26]. However, Delaunay triangulation may not be boundary conforming. Moreover, unlike

two-dimensional case, sometime constrained Delaunay triangulation for given boundary triangulation does not exist. The solution is to add new points along the boundary to block the penetration of tetrahedra through the boundary surface. However, according to Ref. 2 this method does not guarantee the desired result for arbitrary case. The other method is a direct division of tetrahedra to recover the boundary surface [26]. Theoretically this method guarantees recovery of boundary surface but we have found that this method cannot be applied to arbitrary surface triangulation and the optimization of surface triangulation is required before the boundary recovering.

Here a "Delaunay-type" approach based on edge swapping method is used. A simple and robust algorithm for the boundary constraining by edge swapping followed by a direct division of tetrahedra is applied.

2.6.2 THREE-DIMENSIONAL EDGE SWAPPING INSERTION

The method for tetrahedral grid generation is based on the three-dimensional edge swapping concept described in [25] and based on Lawson theorem [27]. There are some advantages of this method over the widely used Watson method. First, this method can produce grids optimized by some mesh quality measure other than the Delaunay circumsphere test. This is important because three-dimensional Delaunay grid can result in poorly shaped tetrahedra. Second, this method can be applied to recover boundary faces. Unfortunately the three-dimensional edge swapping may result in getting stuck in local optima. As a result, the recovering of boundary faces is not always successful and other method should be applied in this case.

Procedure of a new node insertion into the current triangulation serves as a basic of the grid generation method.

For each node to be introduced, the tetrahedron containing the node is located. This is typically an efficient local search starting with some tetrahedron. The lately inserted tetrahedron can be used as a starting one. Three cases can be realized.

1. The node is inside of the tetrahedron, see Fig. 2.26a, then the tetrahedron is divided into four.
2. The node belongs to a face of two adjacent tetrahedra, see Fig. 2.26b, in this case the both tetrahedra are divided into three.
3. The node belongs to an edge of the tetrahedron. In this case the node belongs to all tetrahedra sharing the edge, see Fig. 2.26c, and all the tetrahedra are divided into two.

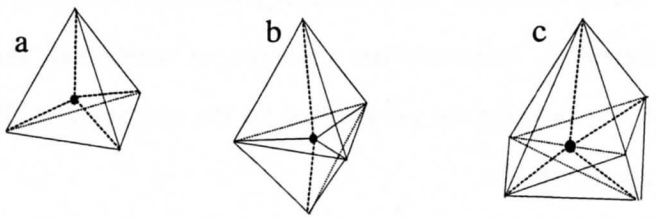


Fig. 2.26 Available cases for node insertion

The faces of the newly created tetrahedra can be swapped to satisfy the optimization criterion. For each face to be swapped we consider 5-node configuration - 3 nodes belonging to the face and 2 nodes opposing to the face. The configuration is swappable if the 5-node configuration is convex, see Ref. 25. The swappable configurations are shown in Fig. 2.27. Three cases can be realized:

1. swapping of 2 tetrahedra forming convex configuration into 3 tetrahedra,
2. swapping of 3 tetrahedra forming convex configuration into 2 tetrahedra,
3. swapping of 2 tetrahedra having two faces belonging to one plane into 2 tetrahedra.

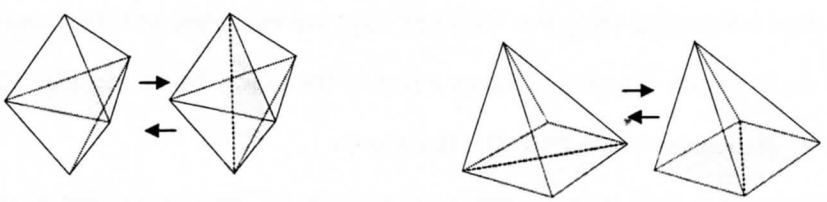


Fig. 2.27 Swappable 5-node configurations

The criterion of swapping is usually Delaunay circumsphere test, but other quality measures are possible. If the circumsphere test is used and the insertion is performed into an existing Delaunay triangulation, the Dealunay grid will be produced [28].

The three-dimensional incremental insertion methods are very sensitive to round-off errors. If the error occurs during the convexity test (computing of tetrahedra volumes), the code may generate tetrahedra with negative volume that results in fatal error in data structure. The use of tolerance thresholds [26] does not produce the desired improvement. We have found that the convexity test should be computed *exactly.* It could be possible if we fix the minimal change of coordinates' value that is the same as transformation of

all coordinates into integer arithmetic. So, if seven digit precision is used for coordinates, 21 digit precision is necessary to compute the convexity test (volumes of tetrahedra). Hopefully the round-off errors appearing during the circumsphere test are not fatal for the resulting grid. This round-off errors treatment results in an absolutely robust code.

2.6.3 GRID GENERATION PROCEDURE

(1) Triangulation of Boundary Points

The procedure starts from the given boundary surface triangulation. Initially the background three-dimensional mesh is established. Usually this mesh involves one tetrahedron containing whole computational domain. All boundary nodes are incrementally inserted into the triangulation by the above edge swapping insertion algorithm. As a result the Delaunay triangulation of the boundary nodes is obtained. As it was mentioned before, this grid does not necessarily contain all boundary faces and the next step is to recover the missing boundary faces.

(2) Boundary Face Recovery

For two dimensions the following algorithm can be used to recover boundaries. All edges from the current triangulation intersecting the given boundary edge are examined and their successive swapping is performed until the boundary segment becomes a part of the triangulation, see Fig. 2.28. In this case the order of edges to swap is important to prevent infinite loops.

Unlike two-dimensional case, three-dimensional problem of face recovering is not straightforward. Sometime the swapping does not recover the desired face. In this case extra nodes should be involved into the triangulation. A direct division of tetrahedra is used to recover the missing faces. This method was proposed originally in Ref. 26. Here another algorithm is proposed which provide the same result but the number of cases to consider is significantly reduced.

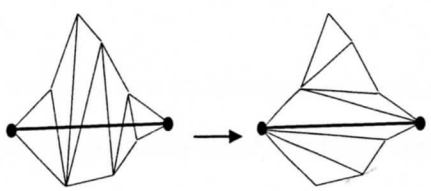


Fig. 2.28 Boundary edge recovery by edge swapping method

Let's consider some boundary face (triangle) which does not exist in the tetrahedral construction. Because all nodes of the face are integrated into the triangulation, the problem is to recover its edges and the face itself. An edge is characterized by two nodes (ends). We start from one of the nodes, let's call it as *A*, see Fig. 2.29, and inspect all tetrahedra having this node as a vertex. If the edge intersects a tetrahedron, we introduce a new point *Q* at the place of intersection. Three possibilities can be realized:

(a) The edge intersects a face of the tetrahedron, see Fig. 2.29a.

(b) The edge intersects an edge of the tetrahedron, see Fig. 2.29b.

(c) The edge coincides with an edge of the tetrahedron, see Fig. 2.29c.

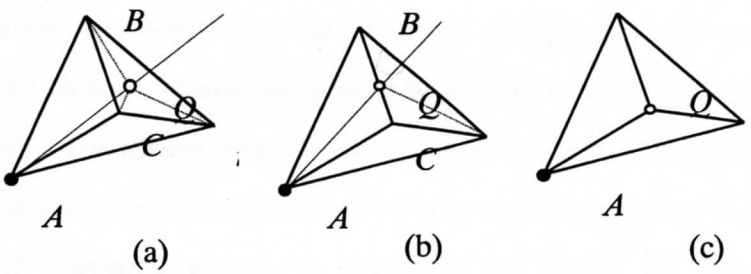


Fig. 2.29 Edge recovery by direct division of tetrahedra

In case (a) the node *Q* is inserted by dividing of two tetrahedra adjacent to the face *BCD* into three, see Fig. 2.26b. In case (b) all tetrahedra having the edge *BC* are divided into two, see Fig. 2.26c. The segment *AQ* is stored as a front segment. If the node *Q* was inserted we repeat the same procedure starting from point *Q* instead of *A*, otherwise the process for the edge is done. As a result the boundary of the considered surface face is represented as a set of line segments which form the initial front for face recovery, see Fig. 2.30.

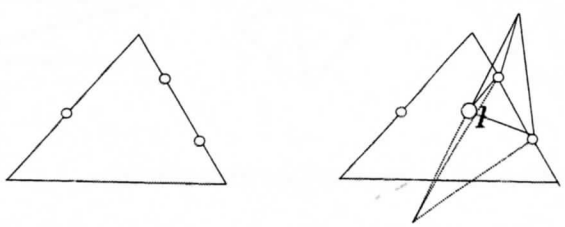


Fig. 2.30 Face recovery by direct division of tetrahedra

The face is recovered by the following advancing front procedure. We choose any segment from the front and examine all tetrahedra having this segment as an edge. If there is a tetrahedron whose face is lying in the front plane we mark this face as a boundary face and update the front. If there is no such tetrahedron, the tetrahedron crossing the face is taken in turn and new node P is inserted by division of the tetrahedron into two as shown in Fig. 2.30 Next front segment is then considered. The process stops after the front becomes empty. Finally, all tetrahedra exterior to the computational domain are removed.

This direct division of tetrahedra should be always successful, however some distorted tetrahedra near boundary surface can be generated. Moreover, the code might be failed because of round-off errors occurred during the determination of the new point position. The reason is in the feature of Delaunay triangulation to connect the closest points. Usually Delaunay tetrahedral grid contains some faces situated along the boundary surface but sometime these faces do not exactly coincide with the given boundary triangles. So, if we try to recover the given surface triangles exactly, some flat tetrahedra might be generated. Figure 2.31 shows an example of such situation. The tetrahedral construction and the surface triangulation shown in Fig. 2.31(a) lead to forming of distorted tetrahedra. The resulting boundary triangulation is shown in Fig. 2.31(b). More complex cases like one shown in Fig. 2.31(c) might also occur. In this Figure the dashed line shows a site from tetrahedral construction. To avoid these situations swapping of boundary triangulation is necessary. One possibility is to optimize the surface triangulation in advance. This method applies edge swapping with Delaunay circumcircle criterion in plane tangential to each pair of triangles, see Ref. 29, and has been successively used in original code. The second method is to swap boundary edges according to the existing three-dimensional triangulation. Not all boundary faces can be recovered by such swapping but the most dangerous cases producing degenerated tetrahedra are removed very efficiently.

(a)bold lines correspond to given surface trian-gulation

(b)surface triangulation obtained by the direct divi-sion of tetrahedra

(c)case involving more than two boundary faces

Fig. 2.31 Cases leading to failure of boundary recovering procedure

(3) Interior Nodes Generation

The interior nodes are generated by the Frontal Delaunay refinement technique similar to one proposed in Ref. [30,31]. This technique does not require tracking of the front.

1. The distribution function is computed for each boundary point. This function is the average of the boundary edge lengths surrounding a point.
2. The active flag is assigned to all tetrahedra. The flag is off if the tetrahedron cannot be refined. This is assumed to occur when its maximal side length is less then the average distribution function for the tetrahedron.
3. All faces are inspected. If the face is an interface between an active tetrahedron and tetrahedron which is turned off, or if the face belongs to the boundary and its neighboring terahedron is active, a new point is created by advancing from the face in a direction normal to the selected face. A distance is determined by the average distribution function for the face.
4. The element containing the new point is searched.
5. The distribution function is interpolated to the new point. The existing points whose distance from the new point is less than 0.7 times the point distribution function are searched. If the new point is too close to an existing point, the new point is rejected and step 3 is performed.
6. Insert new point by the insertion algorithm described above. Change the active flag to all newly created or altered tetrahedra.
7. Repeat steps 3-6 until no more points can be inserted.
8. Optimize the mesh.

2.7 格子生成法の適用例

2.7.1 NAL 超音速実験機モデル

この NAL の超音速旅客機の実験機モデルは，前章で表面格子生成過程を説明する例としてすでに取り上げている．この実験機自体には推進装置が付いておらず，固体ロケットブースターを取り付けて高度 15,000m まで打ち上げてからブースターを切り離し，機体を滑空させて空力特性のデータを収集する実験が 2002 年に計画されている．この格子は，ブースター切り離し時の空力干渉を評価するために作成された．

図 2.13 に示される機体形状は，CATIA を用いて作成され，STL 書式で出力されたものを表示している．この STL 表現では，約 27,000 個の三角形セルを用いている．図 2.14 に示される稜線構築過程を経て，図 2.19 示さている表面格子生成を行った．表面格子の節点数は 100,044，三角形セル数は 200,084 であった．次に，図 2.24 に示される外部境界を作成し，空間格子を生成している．

2.7.2 全翼機モデル

図 2.32 の全翼機（BWB; Blended-Wing-Body）モデルの STL データは CATIA を用いて製作された．稜線の自動構築後に，エンジンナセル部分に格子密度制御を行うために数本の稜線を追加した（図 2.33 参照）．その後，図 2.34 に示したように表面格子生成が行われた．

この表面格子は，節点数 46,933 個，セル数 93,431 個であった．この例では Pentium III 600MHz を搭載したパソコンを用いて，初期フロントを生成した後の格子生成に要した CPU 時間は約 5 分であった．また，表面格子生成全体に要した時間は約 1 時間半であった．この時間の大半は，それぞれの稜線において分割パラメータを試行錯誤で決める作業に費やされた．しかし一度格子生成を行えば，初期フロント作成前までの作業内容すべてをファイルに保存しておくことで，二回目以降の表面格子生成はこのファイルから作業を開始することにより，例えば前縁部分の稜線の分割数だけ変更したい場合などには簡単に新たな表面格子を作成することができる．

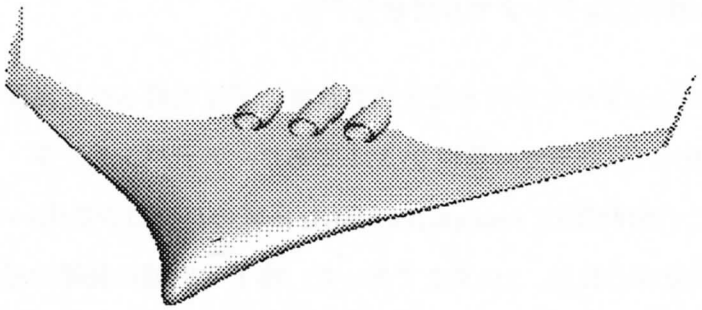


図 2.32　全翼機モデル

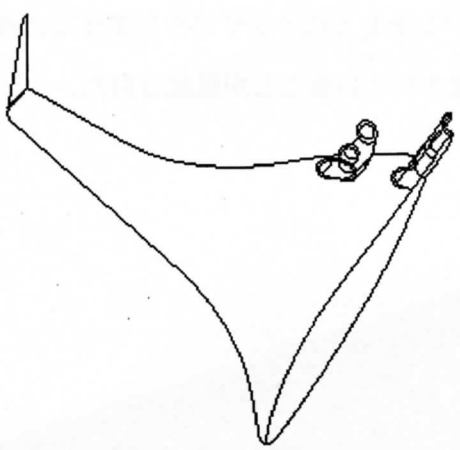


図 2.33 全翼機モデルに対する稜線の構築

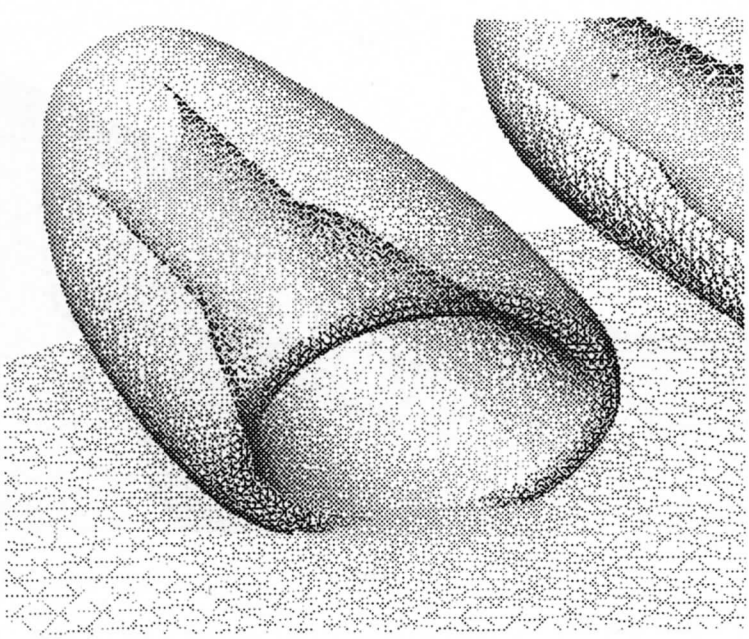
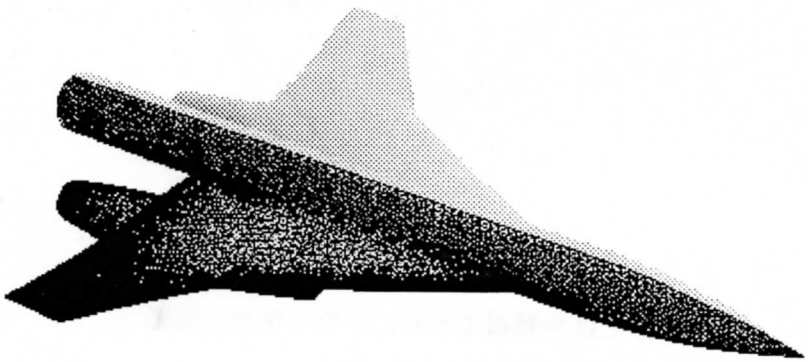


図 2.34 全翼機モデルの表面格子

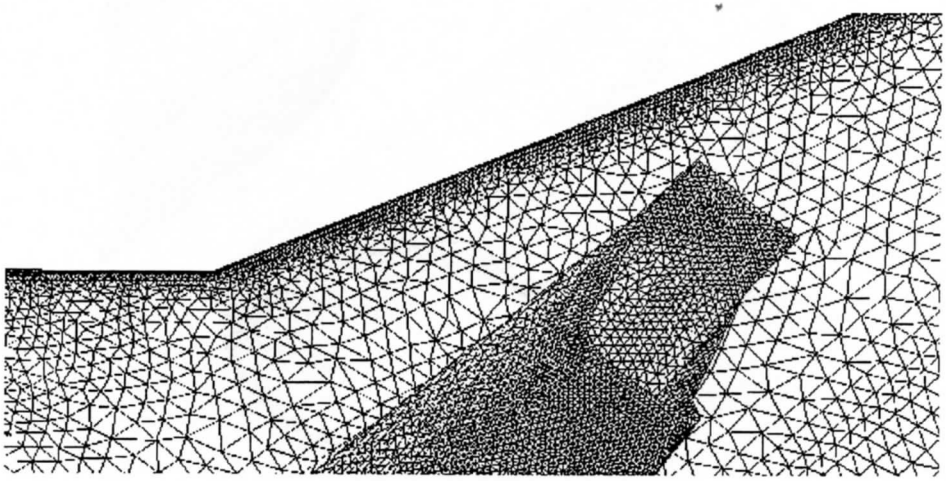
2.7.3 NAL 超音速実験機のエンジンナセル付きモデル

図 2.35 は NAL のエンジンナセル付き超音速実験機モデルで得られた表面格子である．この表面格子の節点数は 45,937，三角形セル数は 91,874 であり，アドバンシング・フロント法を用いた格子生成自体に要した CPU 時間は，Pentium II 450MHz 搭載のパソコンを用いて約 10 分であった．また空間格子生成後の総節点数は，256,802 であった．格子生成前の稜線の追加や削除，また分割数や格子点分布を指定するといった作業に要した時間は，この場合 1 時間程度であった．

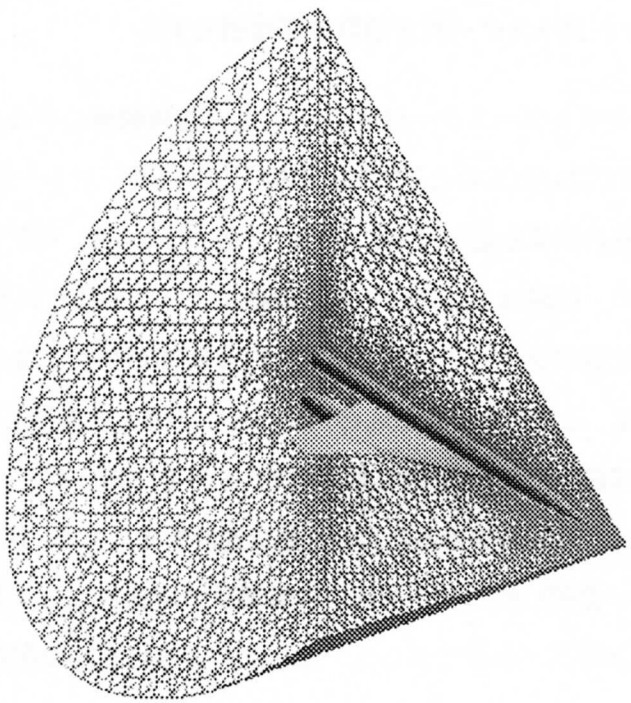
図 2.36 は Euler 計算の結果をもとに作成した表面圧力分布である．計算条件は，自由流マッハ数 2.0，迎角 0 度である．エンジンナセルのインテイクで発生した衝撃波がよく捉えられ，本手法により質のよい表面格子が生成されていることが確認された．



(a)



(b)



(c)

図 2.35 NAL エンジンナセル付き超音速実験機モデルの表面格子:(a) 機体表面, (b) 翼下面から見た表面格子, (c)外部境界付きの表面格子

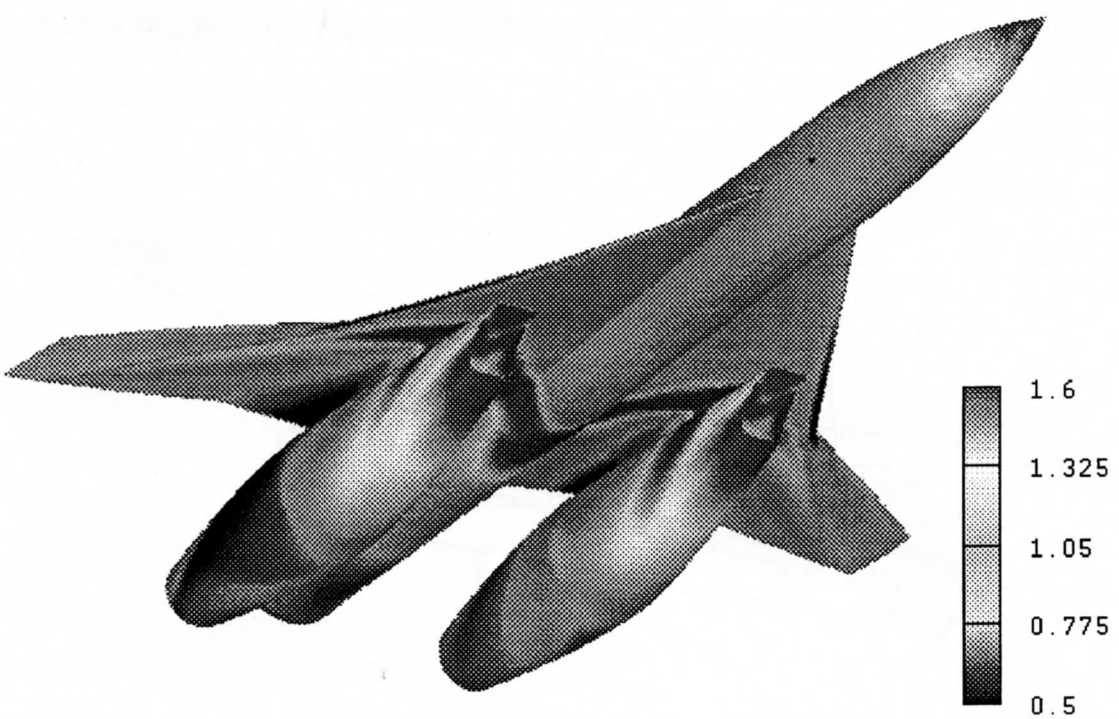


図 2.36 表面圧力分布（M = 2.0, α = 0.0ú

2.7.4 NAL 固体ロケットブースター付き超音速実験機モデル

図2.37-2.42に固体ロケットブースター付きのNAL超音速実験機モデルのCFD計算過程を示す．この機体自体は2.7.1節で用いたものと基本的には同じであるが，ピトー管などの詳細な部品が取り付けられている．稜線の構築と追加は前に述べた方法で行われた（図 2.37 参照）．このモデルは非常に複雑であるため，稜線は501本，これら稜線によって形成される閉領域は213個に及んだ．また稜線の追加と分割パラメータの指定作業に約4時間が必要となった．得られた表面格子は，格子点数105,112個，三角形セル数208,497個であり（図2.38参照），Pentium III 600MHz搭載のパソコンで格子生成自体に要したCPU時間は約35分をであった．

この事例ではよりよい計算結果を得るために，遷音速計算用（半球；図2.39a参照）と超音速形算用（半円錐；図 2.39b 参照）に二つの形状の外部境界が用意された．これら外部境界の生成には，各々10分程度の追加時間が必要であった．さらに四面体分割での空間格子生成がそれぞれ行われ，外部境界が半球形状のものは格子点数658,532個，四面体要素数3,570,327個，また半円錐形状のものは格子点数758,003個，四面体要素数4,158,292個となった．

それぞれの空間格子を元にEuler計算が行われた．図2.40-2.35にそれぞれの結果から作成した圧力分布図を示す．これらの図より，各々の部品での衝撃波のおおよそは捉えられているが，格子点量が十分でないため滑らかになっていないことが分かる．今後，より詳細な格子での計算が望まれる．

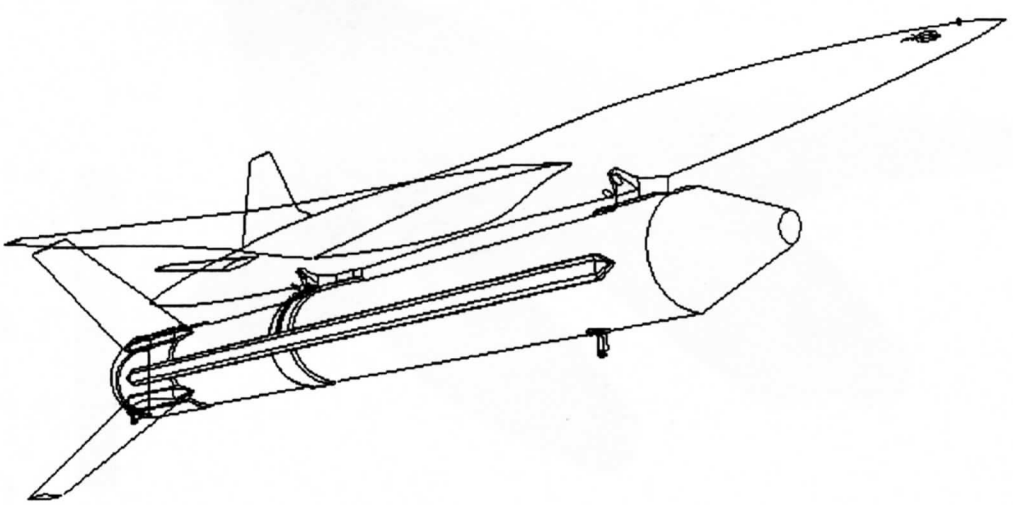
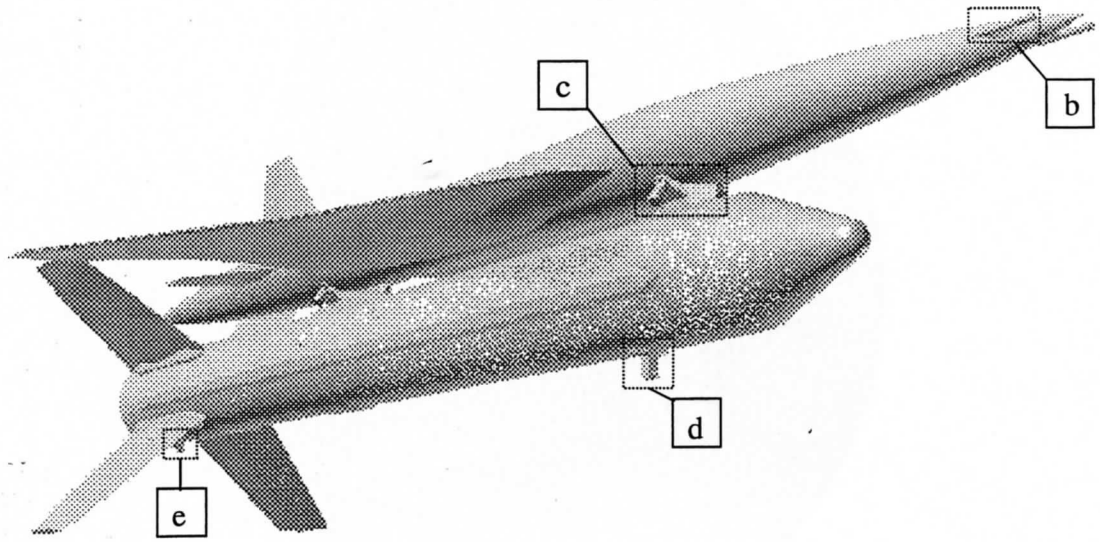
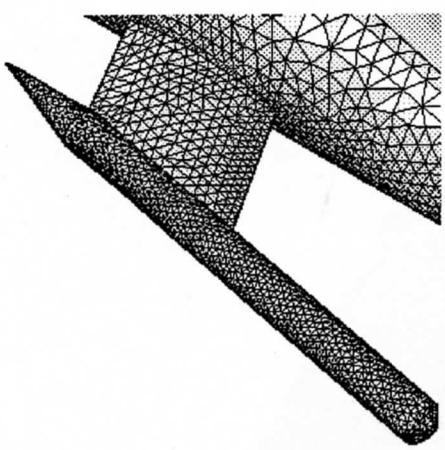


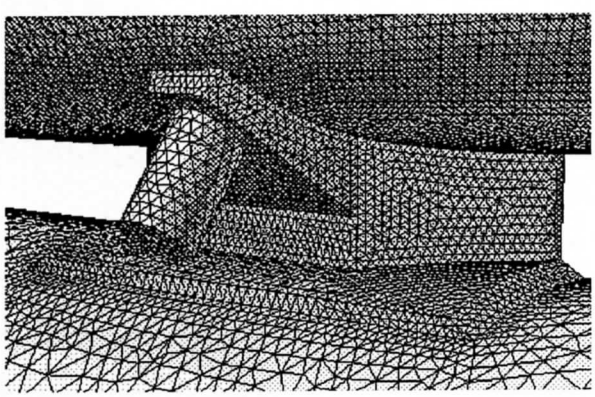
図 2.37 NAL固体ロケットブースター付き超音速実験機モデルの稜線



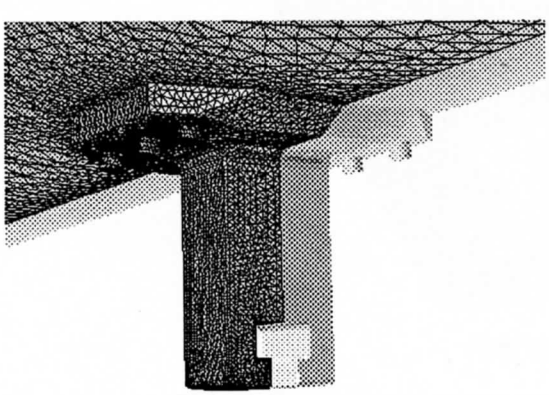
(a) 全体像



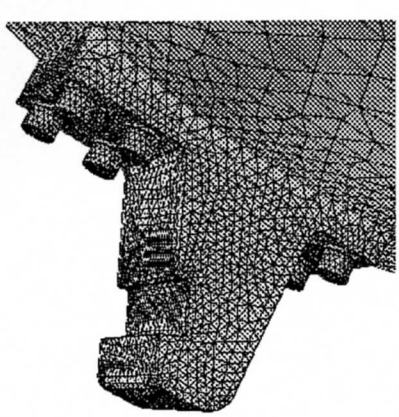
(b) ピトー管



(c) 前部結合部

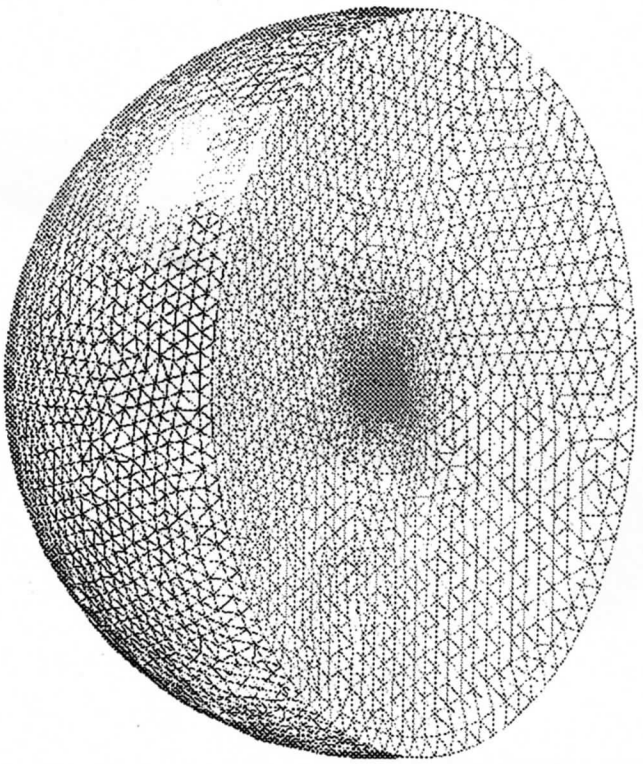


(d) 前部取り付け金具

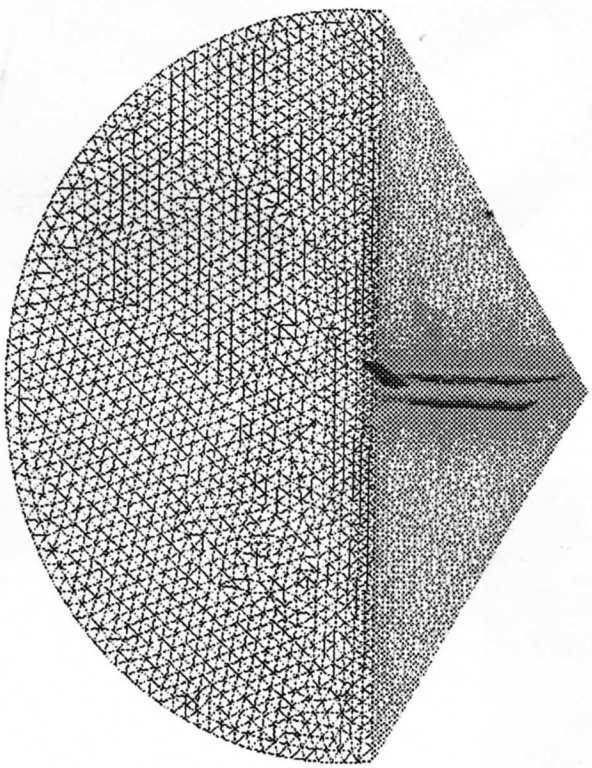


(e) 後部取り付け金具

図 2.38 NAL 固体ロケットブースター付き超音速実験機モデルの表面格子

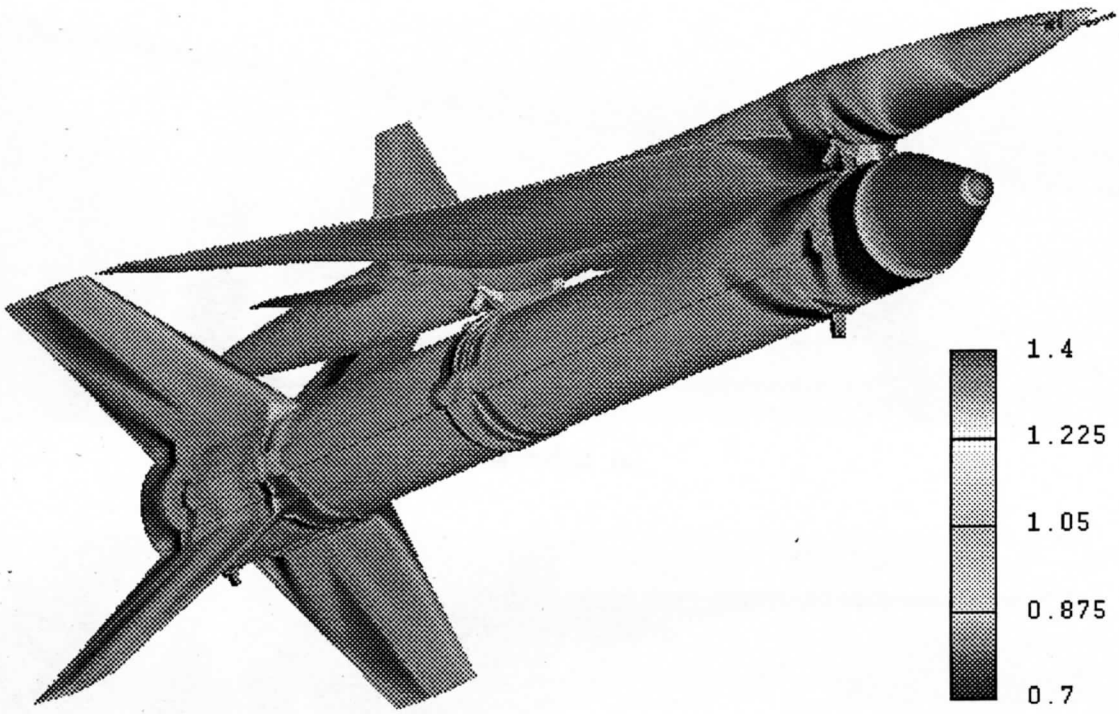


(a) 遷音速計算用

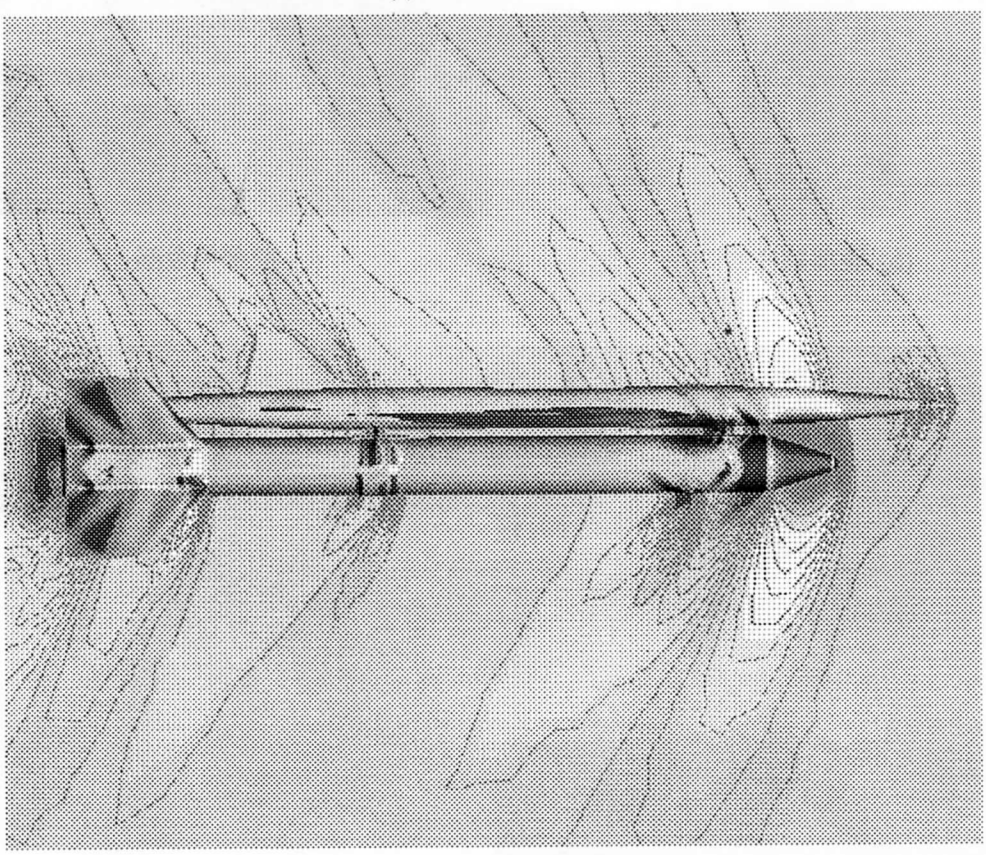


(b) 超音速計算用

図 2.39 NAL 固体ロケットブースター付き超音速実験機モデルの外部境界

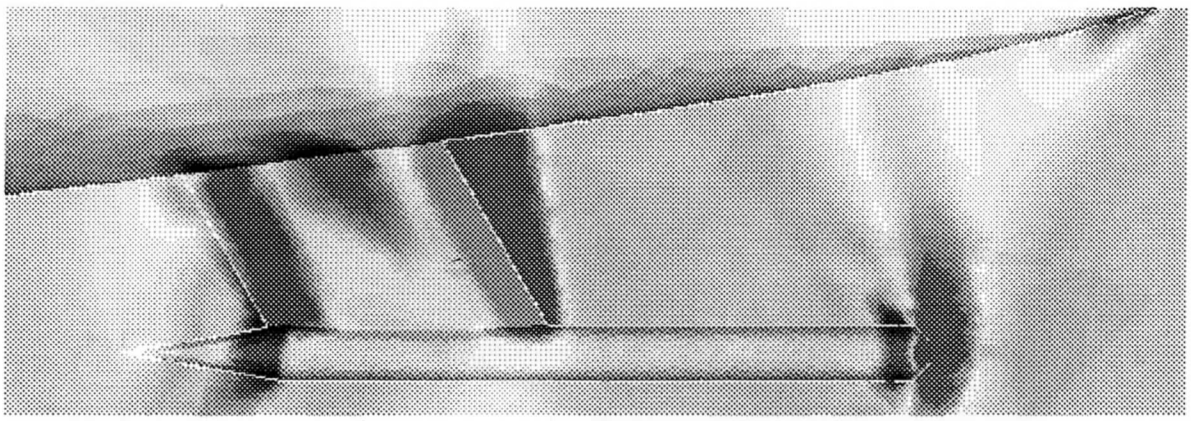


(a) 表面圧力分布

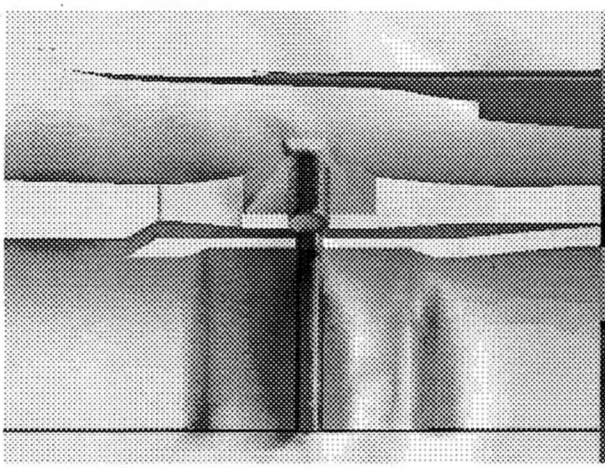


(b) 対称面での圧力分布

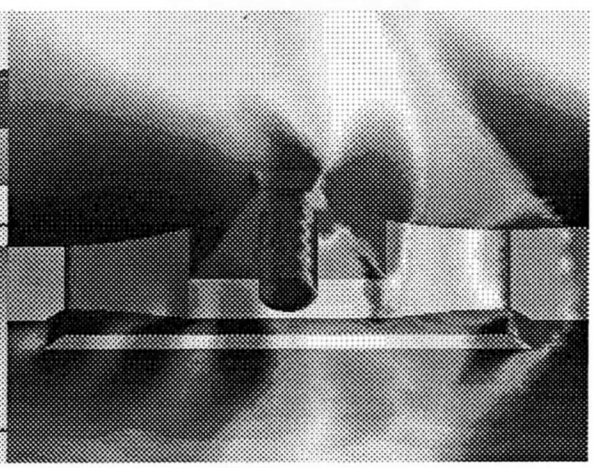
図 2.40　NAL 固体ロケットブースター付き超音速実験機モデルの遷音速用格子（図 2.39a）を用いた Euler 計算例（$M = 1.2, \alpha = 0.0$û



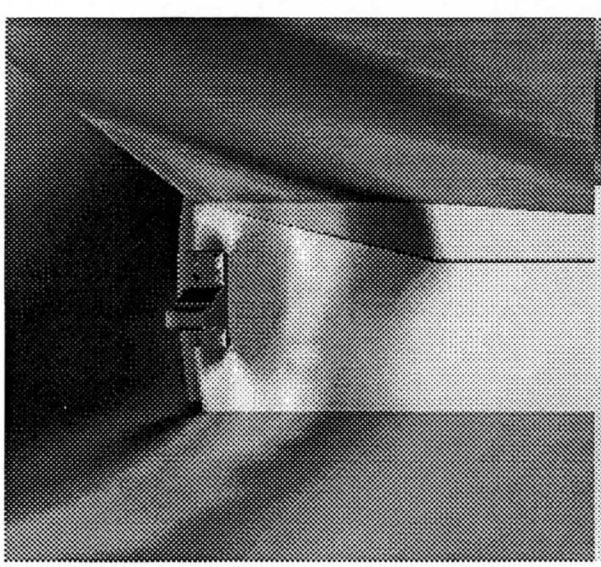
(a) ピトー管



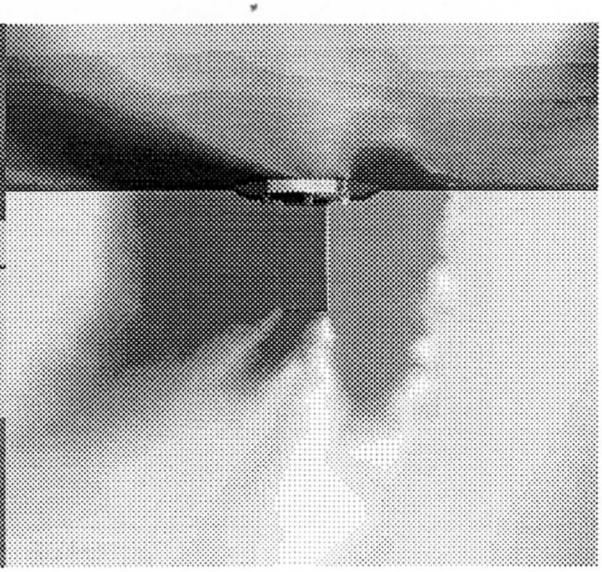
(b) 後部結合部



(c) 前部結合部

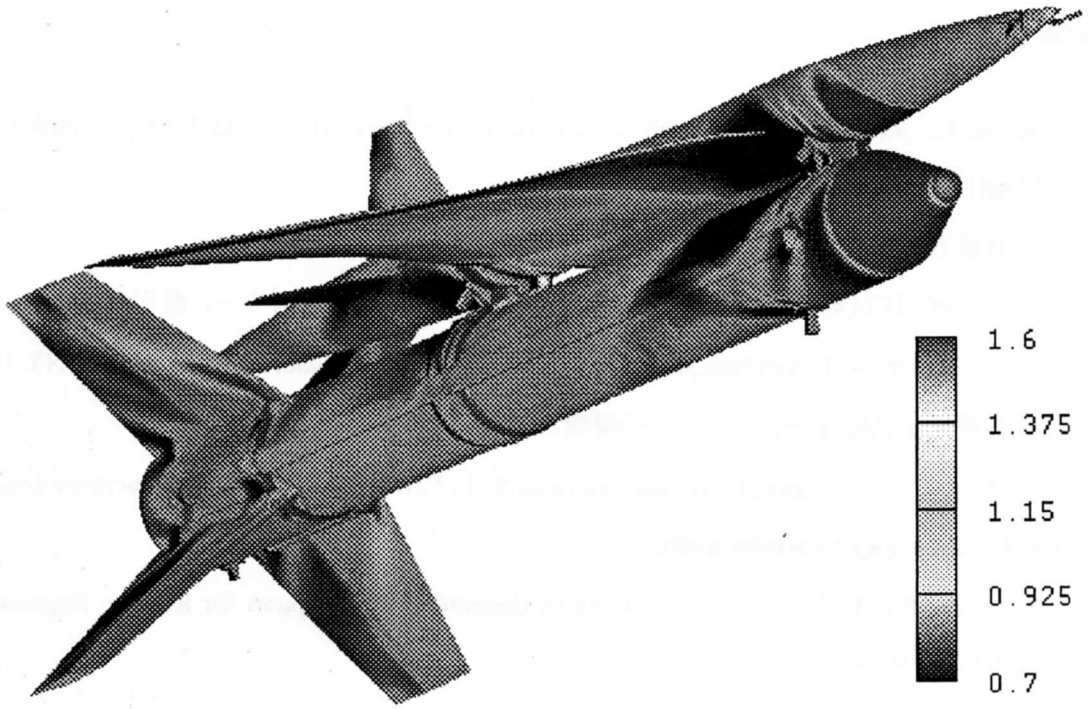


(d) 後部取り付け金具

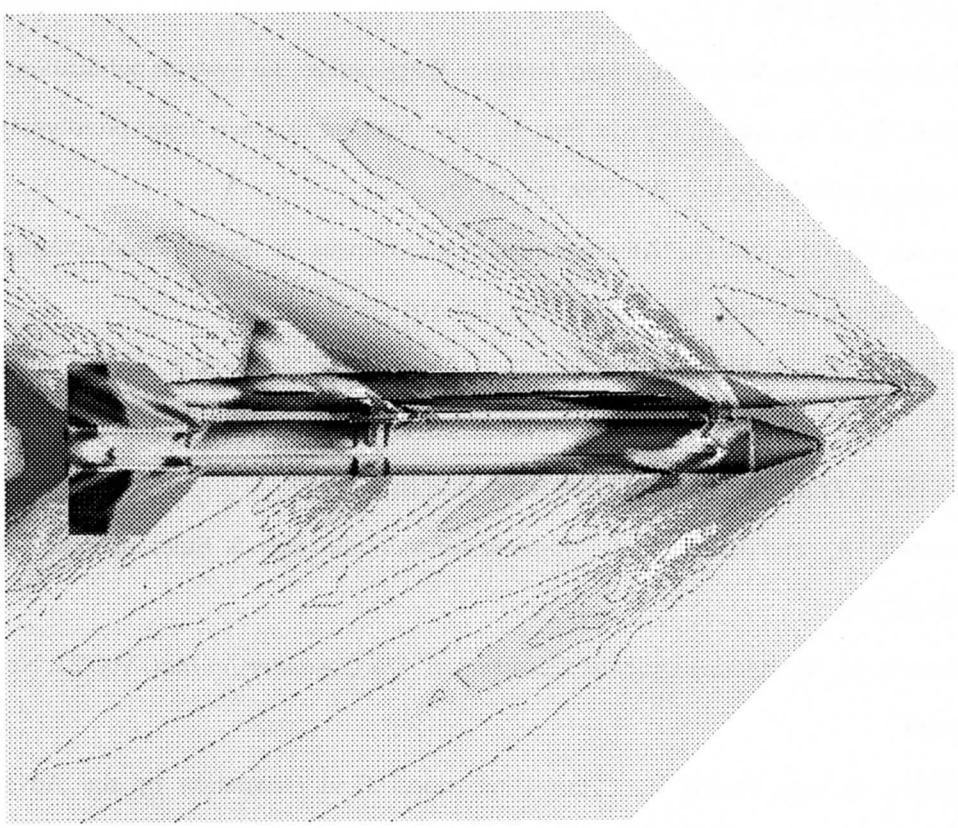


(e) 前部取り付け金具

図 2.41 $M = 1.2, \alpha = 0.0$úのときの圧力分布（図 2.40）の詳細図



(a) 表面圧力分布



(b) 対称面での圧力分布

図 2.42　NAL 固体ロケットブースター付き超音速実験機モデルの超音速用格子（図 2.39b）を用いた Euler 計算例（M = 2.0, α = 0.0û

参考文献

[1] Kania, L., and Warsi, S., "Curvature Adapted Triangulation of NURBS Surface," AIAA Paper 97-1981-CP, 1997.

[2] 早野龍五, 高橋忠幸, *計算物理*, 共立出版, 東京, 1992.

[3] 桜田幸嗣, 田口景介, *Visual C++5.0 プログラミング入門*, アスキー, 東京, 1997.

[4] Neider, J., Davis, T., and Woo, M., *OpenGL™ Programming Guide*, Addison Wesley, NY, 1993.

[5] 日本機械学会編, *CAD システムの機能と構成*, 技報堂出版, 東京, 1987.

[6] Hufford, G. S., Mitchell, C. R., and Harrand, V. J., "Trimmed NURBS, Unstructured Grids and CFD," AIAA Paper 96-1996, 1996.

[7] Ladeinde, F., "Truly Automatic CFD Mesh Generation with Support for Reverse Engineering," AIAA Paper 99-0828, 1999.

[8] Woan, C. J., "Unstructured Surface Grid Generation on Unstructured Quilt of Patches," AIAA Paper 95-2202, 1995.

[9] 3D Systems, Inc., *Stereolithography Interface Specification*, 3D Systems publication, CA, 1989.

[10] Jacob, G. G. K., Kai, C. C., and Mei, T., "Development of a new rapid prototyping interface," *Computers in Industry*, Vol. 39, 1999, pp. 61-70.

[11] Kumar, V., and Dutta, D., "An assessment of data formats for layered manufacturing," *Advances in Engineering Software*, Vol. 28, 1997, pp. 151-164.

[12] 中橋和博, 藤井孝藏, *格子生成法とコンピュータグラフィックス*, 東京大学出版会, 東京, 1995.

[13] 谷口健男, *FEM のための要素自動分割*, 森北出版, 1992.

[14] Jameson, A., and Baker, T. J., "Improvements to the aircraft Euler method," AIAA Paper 87-0425, 1987.

[15] Bowyer, A., "Computing Dirichlet tessellations," *The Computer Journal*, Vol. 23, No. 2, 1981, pp. 162-166.

[16] Löhner, R., and Parikh, P., "Three-Dimensional Grid Generation by the Advancing Front Method," *International Journal for Numerical Methods in Fluids*, Vol. 8, 1988, pp. 1135-1149.

[17] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewiez, P. C., "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, Vol. 72, 1987, pp. 449-466.

[18] Morgan, K., Peraire, J. and Peiró, J., "Unstructured grid methods for compressible flows," in AGARD Report 787, Special Course on Unstructured Grid Methods for Advection Dominated

Flows, 1992, 5.1-5.39.

[19] Löhner, R., "Extending the Range of Applicability and Automation of the Advancing Front Grid Generation Technique," AIAA Paper 96-0033, 1996.

[20] Nakahashi, K., and Sharov, D., "Direct Surface Triangulation Using the Advancing Front Method," AIAA Paper 95-1686-CP, 1995, pp.442-451.

[21] Iwamiya, T., "NAL SST Project and Aerodynamic Design of Experimental Aircraft," to be published in Proceedings of the 4th ECCOMAS Computational Fluid Dynamics Conference, Athens, John Wiley & Sons Ltd., Chichester, 1998, pp.580-585.

[22] Vinokur, M., "On one-dimensional stretching functions for finite-difference calculations," *Journal of Computational Physics*, Vol. 50, No. 2, 1983, pp.215-234.

[23] Löhner, R., "Regridding Surface Triangulation," *Journal of Computational Physics*, Vol. 126, No. 1, 1996, pp.1-10.

[24] Sharov, D. and Nakahashi, K., "A Boundary Recovery Algorithm for Delaunay Tetrahedral Meshing," 5th International Conference on Numerical Grid Generation in Computational Field Simulations, 1996, pp.229-238.

[25] T.J. Barth, Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations, in Special Course on Unstructured Grid Methods for Advection Dominated Flows, AGARD-R-787, 1992, pp. 6-1 - 6-61.

[26] N.P. Weatherill, O. Hassan, Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints, *Int.J. for Num.Meth.Engng.*, **37**, 2005-2039 (1994).

[27] C.L. Lawson, Properties of n-dimensional Triangulations, CAGD, **3,** 231-246, (1986).

[28] V.T. Rajan, Optimality of the Delaunay Triangulation in R^d, Proc. of the 7th ACM Symp. on Computational Geometry, 1991, p.p.357-372.

[29] K. Nakahashi, D. Sharov, Direct Surface Triangulation Using the Advancing Front Method, AIAA 12th CFD Conference, AIAA 95-1686, 442-451, (1995).

[30] J.-D. Müller, Quality Estimates and Stretched Meshes Based on Delaunay Triangulatiuons, *AIAA J.*, **32**, 2372-2379, (1994).

[31] D.L. Marcum, Generation of Unstructured Grids for Viscous Flow Applications, AIAA Paper 95-0212, (1995).

第３章　非構造格子上での Euler/Navier-Stokes 方程式数値解法

3－1 支配方程式

支配方程式として用いる三次元非定常 Navier-Stokes 方程式は積分形表示で以下のようになる．

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{Q}dV+\int_{\partial\Omega}\left[\mathbf{F}(\mathbf{Q})-\mathbf{G}(\mathbf{Q})\right]\cdot\mathbf{n}dS=0 \tag{3.1}$$

$\mathbf{Q}$は保存量ベクトルで，その成分は

$$\mathbf{Q}=\left[\rho,\rho u,\rho v,\rho w,e\right]^{T} \tag{3.2}$$

である．ここで，ρは密度，u，v，wはそれぞれx，y，z方向の速度成分，eは全エネルギーである．また，$\mathbf{n}$は検査体積Ωの境界面$\partial\Omega$における検査体積の外向き法線ベクトルである．$\mathbf{F}(\mathbf{Q})$，$\mathbf{G}(\mathbf{Q})$はそれぞれ非粘性流束ベクトル，粘性流束ベクトルであり，$\mathbf{i}$，$\mathbf{j}$，$\mathbf{k}$をx，y，z方向の単位ベクトルとすると，以下のように表わされる．

$$\mathbf{F}(\mathbf{Q})=\begin{bmatrix}\rho u\\ \rho u^{2}+p\\ \rho uv\\ \rho uw\\ (e+p)u\end{bmatrix}\mathbf{i}+\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^{2}+p\\ \rho vw\\ (e+p)v\end{bmatrix}\mathbf{j}+\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^{2}+p\\ (e+p)w\end{bmatrix}\mathbf{k} \tag{3.3}$$

$$\mathbf{G}(\mathbf{Q})=\begin{bmatrix}0\\ \tau_{xx}\\ \tau_{xy}\\ \tau_{xz}\\ g_{x}\end{bmatrix}\mathbf{i}+\begin{bmatrix}0\\ \tau_{xy}\\ \tau_{yy}\\ \tau_{yz}\\ g_{y}\end{bmatrix}\mathbf{j}+\begin{bmatrix}0\\ \tau_{xz}\\ \tau_{yz}\\ \tau_{zz}\\ g_{z}\end{bmatrix}\mathbf{k},$$

$$g_{x}=\tau_{xx}u+\tau_{xy}v+\tau_{xz}w-q_{x},\quad g_{y}=\tau_{xy}u+\tau_{yy}v+\tau_{yz}w-q_{y},$$
$$g_{z}=\tau_{xz}u+\tau_{yz}v+\tau_{zz}w-q_{z} \tag{3.4}$$

ここで，pは圧力，τ，qはそれぞれ応力テンソル，熱流束で

$$\tau_{xx}=\frac{2}{3}\mu\left(2\frac{\partial u}{\partial x}-\frac{\partial v}{\partial y}-\frac{\partial w}{\partial z}\right),\quad \tau_{yy}=\frac{2}{3}\mu\left(2\frac{\partial v}{\partial y}-\frac{\partial u}{\partial x}-\frac{\partial w}{\partial z}\right),\quad \tau_{zz}=\frac{2}{3}\mu\left(2\frac{\partial w}{\partial z}-\frac{\partial u}{\partial x}-\frac{\partial v}{\partial y}\right),$$

$$\tau_{xy}=\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right),\quad \tau_{xz}=\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right),\quad \tau_{yz}=\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right) \tag{3.5}$$

$$q_x = -k\frac{\partial T}{\partial x},\quad q_y = -k\frac{\partial T}{\partial y},\quad q_z = -k\frac{\partial T}{\partial z} \tag{3.6}$$

となる．Tは温度で，μ，kは粘性係数及び熱伝導係数である．また，$k = \mu C_p/\mathrm{Pr}$(C_pは定圧比熱，Prはプラントル数)と完全気体の状態方程式$p = \rho RT$(Rは空気の気体定数)の関係から熱流束qは以下のように表わされる．

$$q_x = -\frac{\gamma}{\gamma-1}\cdot\frac{\mu}{\mathrm{Pr}}\cdot\frac{\partial}{\partial x}\left(\frac{p}{\rho}\right),\quad q_y = -\frac{\gamma}{\gamma-1}\cdot\frac{\mu}{\mathrm{Pr}}\cdot\frac{\partial}{\partial y}\left(\frac{p}{\rho}\right),$$

$$q_z = -\frac{\gamma}{\gamma-1}\cdot\frac{\mu}{\mathrm{Pr}}\cdot\frac{\partial}{\partial z}\left(\frac{p}{\rho}\right) \tag{3.7}$$

ここで，γは比熱比で理想気体を仮定して 1.4 の一定値とする．式(3.1)の連立方程式は以下の完全気体の状態方程式により閉じられる．

$$p = (\gamma-1)\left[e - \frac{1}{2}\rho\left(u^2+v^2+w^2\right)\right] \tag{3.8}$$

流れが層流の場合，層流のプラントル数Pr_lは 0.72 とする．また，層流の粘性係数μ_lは次のサザーランドの式を用いて求める．

$$\frac{\mu_l}{\mu_0} = \left(\frac{T}{T_0}\right)^{3/2}\frac{T_0+S}{T+S} \tag{3.9}$$

ここで，μ_0は温度T_0における粘性係数で，Sは空気の場合 111(K)の値をとる．

実際の計算では各物理量に以下のような無次元化を施す．

$$\tilde{t} = \frac{t}{L\gamma^{1/2}/a_\infty},\quad \tilde{x} = \frac{x}{L},\quad \tilde{y} = \frac{y}{L},\quad \tilde{z} = \frac{z}{L},$$

$$\tilde{u} = \frac{u}{a_\infty/\gamma^{1/2}},\quad \tilde{v} = \frac{v}{a_\infty/\gamma^{1/2}},\quad \tilde{w} = \frac{w}{a_\infty/\gamma^{1/2}}, \tag{3.10}$$

$$\tilde{\rho} = \frac{\rho}{\rho_\infty},\quad \tilde{p} = \frac{p}{p_\infty},\quad \tilde{e} = \frac{e}{p_\infty},\quad \tilde{\mu} = \frac{\mu}{\mu_\infty}$$

添え字の∞は主流での各物理量の値を示しており，Lは代表長さ，aは音速である．これらの変数を式(3.1)に代入して整理し，～を簡略化のため省略して表わすと，

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{Q}dV + \int_{\partial\Omega}\left[\mathbf{F}(\mathbf{Q}) - \frac{1}{\mathrm{Re}}\mathbf{G}(\mathbf{Q})\right]\cdot\mathbf{n}dS = 0 \tag{3.11}$$

無次元化で用いられた主流の各変数などは以下のように表わされるレイノルズ

数 Reにまとめられる.

$$\mathrm{Re} = \frac{\rho_\infty a_\infty L}{\gamma^{1/2}\mu_\infty} = \mathrm{Re}_{ex} \cdot \frac{1}{\gamma^{1/2} M_\infty} \tag{3.12}$$

Re_{ex}は一般的なu_∞を代表的な流速として用いたレイノルズ数で，Mはマッハ数である.

3－2　セル節点有限体積法による離散化

式(3.11)は有限体積法のセル節点法によって離散化される. セル節点法では各格子節点周りに検査体積が構築されるが，その検査体積には，セルの中心，面の中心，辺の中点を結んでできる多面体として定義される非重合二重格子(non-overlapping dual cell)と呼ばれるものを採用する. 図 3.4 にある四面体セルに含まれる辺の両端の節点i，j間に形成される検査体積の境界面を示す. 図中の A は辺の中点，B, D はその辺に接する四面体の面の重心，C は四面体の重心である. 点 A，B，C，D で囲まれる面がその四面体に属する境界面となる.

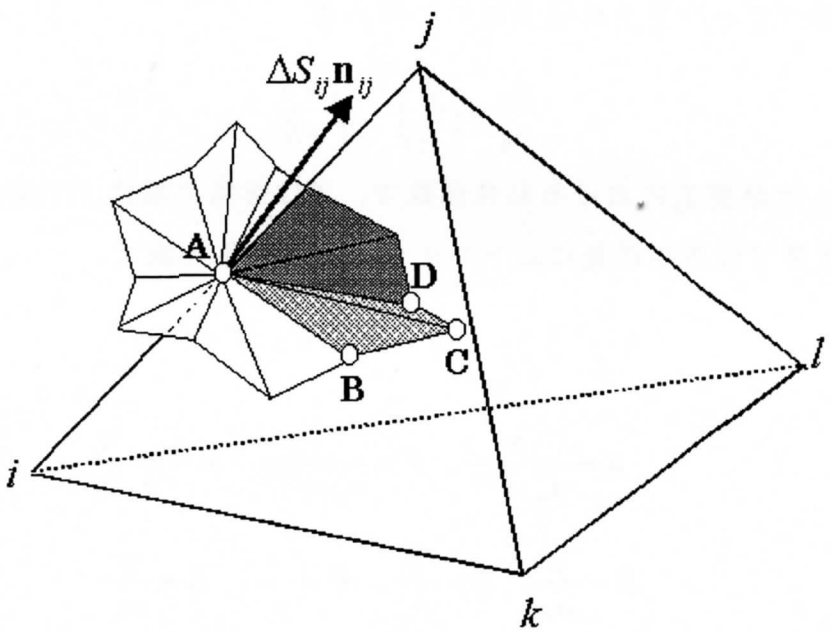


図 3.1 四面体における検査体積境界面

この検査体積において式(3.11)は空間離散化され，次の代数方程式に置き換えられる.

$$\frac{\partial \mathbf{Q}_i}{\partial t} = -\frac{1}{V_i}\left[\sum_{j(i)} \Delta S_{ij}\mathbf{F}(\mathbf{Q})_{ij}\cdot\mathbf{n}_{ij} - \frac{1}{\mathrm{Re}}\sum_{j(i)} \Delta S_{ij}\mathbf{G}(\mathbf{Q})_{ij}\cdot\mathbf{n}_{ij}\right] \tag{3.13}$$

ここで，V_iは節点i周りの検査体積の体積，ΔS_{ij}と$\mathbf{n}_{ij}$はそれぞれ点iと点iに隣接する点jとの間の検査体積境界面の面積と，その単位法線ベクトル(点iから見て外向きを正とする)を示している．また，$\sum_{j(i)}$は点i周りの検査体積を構成する多面体の面の総和を意味している．図3.4のセル内では三角形ABCと三角形ACDの２つの面が検査体積境界面となり，２つの面ベクトルを持つことになる．従って面ベクトル$\Delta S_{ij}\mathbf{n}_{ij}$は点$i$と点$j$を両端とする辺を共有する全てのセル内の２つの面ベクトルを足し合せることによって求められる．

式(3.13)右辺第一項の流束評価には，近似リーマン解法を用いたスキームを適用する．ここでは，風上法の一つであるHLLEW(Harten-Lax-Van Leer-Einfeldt-Wada)法[1]を用いる．すなわち，

$$\mathbf{F}(\mathbf{Q})\cdot\mathbf{n} = \frac{1}{2}\left[\mathbf{f}(\mathbf{Q}_L,\mathbf{n}) + \mathbf{f}(\mathbf{Q}_R,\mathbf{n}) - \left|\mathbf{A}\left(\tilde{\mathbf{Q}},\mathbf{n}\right)\right|\left(\mathbf{Q}_R - \mathbf{Q}_L\right)\right] \tag{3.14}$$

となる．ただし，$\mathbf{f}(\mathbf{Q},\mathbf{n}) = \mathbf{F}(\mathbf{Q})\cdot\mathbf{n}$(数値流束ベクトル)，$\mathbf{A} = \partial\mathbf{f}/\partial\mathbf{Q}$(ヤコビ行列)である．また，添え字$L$，$R$は検査体積境界面の両側における物理量を，～はRoe平均を表わしている．$\left|\mathbf{A}(\tilde{\mathbf{Q}},\mathbf{n})\right|$を求める方法は文献[1]に詳しく書かれている．ここでは，$\mathbf{F}\cdot\mathbf{n}$の最終形を示す．

$$\mathbf{F}\cdot\mathbf{n} = \frac{1}{2}\left[\left(f_1\right)_L\begin{bmatrix}1\\u\\v\\w\\H\end{bmatrix}_L + \left(f_1\right)_R\begin{bmatrix}1\\u\\v\\w\\H\end{bmatrix}_R + \begin{bmatrix}0\\\bar{p}n_x\\\bar{p}n_y\\\bar{p}n_z\\\delta_5\end{bmatrix}\right] \tag{3.15}$$

ここでエンタルピーHは

$$H = (e+p)/\rho \tag{3.16}$$

となり，

$$\left(f_1\right)_L = \rho_L\left(U_L + \hat{\lambda}_1\right) + \delta_1,\quad \left(f_1\right)_R = \rho_R\left(U_R - \hat{\lambda}_1\right) + \delta_1,$$

$$U = un_x + vn_y + wn_z,\quad \bar{p} = p_L + p_R + \delta_2,$$

$$\delta_1 = -\left(\hat{\lambda}^+\Delta p/\tilde{a} + \hat{\lambda}^-\tilde{\rho}\Delta U\right)/2\tilde{a},\quad \delta_2 = -\left(\hat{\lambda}^+\tilde{\rho}\Delta U + \hat{\lambda}^-\Delta p/\tilde{a}\right),\quad \delta_5 = \hat{\lambda}_1\Delta p + \tilde{U}\delta_2,$$

$$\Delta\rho = \rho_R - \rho_L,\quad \Delta p = p_R - p_L,\quad \Delta U = U_R - U_L,$$

$$\hat{\lambda}^+ = \left(\hat{\lambda}_2 + \hat{\lambda}_3\right)\Big/2 - \hat{\lambda}_1,\quad \hat{\lambda}^- = \left(\hat{\lambda}_2 - \hat{\lambda}_3\right)\Big/2 \quad ,$$

$$\begin{bmatrix}\hat{\lambda}_1\\ \hat{\lambda}_2\\ \hat{\lambda}_3\end{bmatrix} = \frac{b_R^+ + b_L^-}{b_R^+ - b_L^-}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} - 2\frac{b_R^+ b_L^-}{b_R^+ - b_L^-}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix} - 2\delta\min\left(b_R^+, b_L^-\right)\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}$$

となる．ただし，λ_1，λ_2，λ_3はヤコビ行列$\mathbf{A}$の固有値で，

$$(\lambda_1,\lambda_2,\lambda_3)^T = (\tilde{U}, \tilde{U}+\tilde{a}, \tilde{U}-\tilde{a})^T$$

となる．また，

$$b_R^+ = \max\left(\tilde{U}+\tilde{a}, U_R + a_R, 0\right),\quad b_L^- = \min\left(\tilde{U}-\tilde{a}, U_L - a_L, 0\right),$$

$$\delta = \min\left(\frac{\rho_{LR}}{|\sigma_1|}, \frac{1}{2}\right),\quad \sigma_1 = \Delta\rho - \Delta p/\tilde{a}^2,\quad \rho_{LR} = \frac{\left(U_L - d_L^-\right)\rho_L + \left(d_R^+ - U_R\right)\rho_R}{d_R^+ - d_L^-},$$

$$d_R^+ = \max\left(\tilde{U}+\tilde{a}, U_R + a_R\right),\quad d_L^- = \min\left(\tilde{U}-\tilde{a}, U_L - a_L\right)$$

である．各物理量のRoe平均は以下のように計算する．

$$\tilde{\rho} = r_L\rho_L + r_R\rho_R,\quad \tilde{H} = r_L H_L + r_R H_R,$$

$$\tilde{u} = r_L u_L + r_R u_R,\quad \tilde{v} = r_L v_L + r_R v_R,\quad \tilde{w} = r_L w_L + r_R w_R,\quad \tilde{U} = \tilde{u}n_x + \tilde{v}n_y + \tilde{w}n_z,$$

$$\tilde{a} = \sqrt{(\gamma-1)\left[\tilde{H} - \frac{1}{2}\left(\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2\right)\right]}$$

ここで，

$$r_L = \frac{\sqrt{\rho_L}}{\sqrt{\rho_L} + \sqrt{\rho_R}},\quad r_R = 1 - r_L = \frac{\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

である．検査体積境界面の両側の物理量L，Rを求める場合，空間一次精度であれば，その面に関する辺の両端の節点での値をそのまま使えばよい．しかし，高次精度にするには検査体積内での基礎物理変数の再構築を行い，境界面での値を両側の節点の値から高次に外挿する必要がある．ここでは節点i周りの検査体積内の基礎物理変数$\mathbf{q} = [\rho, u, v, w, p]^T$はその勾配$\nabla\mathbf{q}_i$を用いて以下のように区分的一次関数で表わされる．

$$\mathbf{q}(\mathbf{r}) = \mathbf{q}_i + \Psi_i \nabla\mathbf{q}_i \cdot (\mathbf{r} - \mathbf{r}_i) \tag{3.17}$$

ここで，$\mathbf{r}$は位置ベクトルである．勾配$\nabla\mathbf{q}_i$は節点iを共有する格子セル内の勾配$\nabla\mathbf{q}_e$を体積平均することによって求める．

$$\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \tag{3.18}$$

$e(i)$は節点iを共有する全ての格子セルを表わし，V_eは格子セルの体積である．セル内の勾配$\nabla \mathbf{q}_e$は四面体格子であれば，図 3.4 の節点i，j，k，lでの値を用いて以下のように求められる．

$$\nabla \mathbf{q}_e = \frac{\partial \mathbf{q}}{\partial x}\mathbf{i} + \frac{\partial \mathbf{q}}{\partial y}\mathbf{j} + \frac{\partial \mathbf{q}}{\partial z}\mathbf{k} \tag{3.19}$$

$$\frac{\partial \mathbf{q}}{\partial x} = \frac{1}{d}\begin{vmatrix} \Delta \mathbf{q}_{ji} & \Delta y_{ji} & \Delta z_{ji} \\ \Delta \mathbf{q}_{ki} & \Delta y_{ki} & \Delta z_{ki} \\ \Delta \mathbf{q}_{li} & \Delta y_{li} & \Delta z_{li} \end{vmatrix}, \quad \frac{\partial \mathbf{q}}{\partial y} = \frac{1}{d}\begin{vmatrix} \Delta x_{ji} & \Delta \mathbf{q}_{ji} & \Delta z_{ji} \\ \Delta x_{ki} & \Delta \mathbf{q}_{ki} & \Delta z_{ki} \\ \Delta x_{li} & \Delta \mathbf{q}_{li} & \Delta z_{li} \end{vmatrix}, \quad \frac{\partial \mathbf{q}}{\partial z} = \frac{1}{d}\begin{vmatrix} \Delta x_{ji} & \Delta y_{ji} & \Delta \mathbf{q}_{ji} \\ \Delta x_{ki} & \Delta y_{ki} & \Delta \mathbf{q}_{ki} \\ \Delta x_{li} & \Delta y_{li} & \Delta \mathbf{q}_{li} \end{vmatrix} \tag{3.20}$$

ここで，$\Delta \mathbf{q}_{ji} = \mathbf{q}_j - \mathbf{q}_i$，$\Delta x_{ji} = x_j - x_i$，$\Delta y_{ji} = y_j - y_i$，…であり，また，

$$d = \begin{vmatrix} \Delta x_{ji} & \Delta y_{ji} & \Delta z_{ji} \\ \Delta x_{ki} & \Delta y_{ki} & \Delta z_{ki} \\ \Delta x_{li} & \Delta y_{li} & \Delta z_{li} \end{vmatrix}$$

となる．一方$\Psi (0 \leqq \Psi \leqq 1)$は流束制限関数と呼ばれるもので，高次精度でスキームの単調性を保持するために用いられる．ここでは，空間精度を悪化させることなく，かつ収束性に優れた Venkatakrishnan の制限関数[2]を適用する．以下にその定義を示す．

$$\Psi_i = \min_{j(i)} \begin{cases} \dfrac{\Delta_{\max}^2 + \varepsilon^2 + 2\Delta_- \Delta_{\max}}{\Delta_{\max}^2 + 2\Delta_-^2 + \Delta_{\max}\Delta_- + \varepsilon^2}, & \text{if } \Delta_- > 0 \\ \dfrac{\Delta_{\min}^2 + \varepsilon^2 + 2\Delta_- \Delta_{\min}}{\Delta_{\min}^2 + 2\Delta_-^2 + \Delta_{\min}\Delta_- + \varepsilon^2}, & \text{if } \Delta_- < 0 \end{cases} \tag{3.21}$$

ここで，

$$\Delta_- = \nabla \mathbf{q}_i \cdot \left(\mathbf{r}_j - \mathbf{r}_i\right)/2, \quad \Delta_{\max} = \max_{j(i)}\left(\mathbf{q}_j - \mathbf{q}_i\right), \quad \Delta_{\min} = \min_{j(i)}\left(\mathbf{q}_j - \mathbf{q}_i\right),$$

$$\varepsilon^2 = \left(K\Delta l\right)^3, \quad K = \text{const}$$

Kは通常 0.1 から 0.3 の値がとられる．Δlはセルの平均長さである．

粘性流束ベクトル$\mathbf{G}(\mathbf{Q})$を計算する際には，その中の応力テンソル及び熱流束に速度と温度の一階導関数が含まれているので，これを検査体積境界面上で評価する必要がある．セル節点法においては，単純に考えるならば各節点上での速度勾

配と温度勾配を計算し，検査体積境界面上での値はその境界面の両端の節点での値の平均をとればよい．しかし文献[3]で指摘されているように，この方法を使うと数値振動が発生する恐れがある．これを防ぐために，各節点上で速度勾配と温度勾配を計算するのではなく，各辺上で直接計算する．つまりqを速度及び温度とすると，その辺上での勾配∇q_{ed}は，

$$\nabla q_{ed} = \frac{1}{V_{ed}} \sum_{e(ed)} \nabla q_e V_e \tag{3.22}$$

ここで，$e(ed)$はその辺を共有する全ての格子セルを表わし，V_{ed}はそれらのセルの体積総和である．

3－3　非構造 LU-SGS 陰解法による時間積分

高レイノルズ数流れの計算では，壁面近傍における細かい計算格子に起因する厳しい安定条件のために，時間積分には陰解法の使用が有利である．ここでは計算時間の大幅な短縮を実現するために，非構造格子法用に拡張された LU-SGS(Lower-Upper Symmetric Gauss-Seidel)陰解法[4]を適用する．

式(3.13)に Euler 陰解法を適用して時間方向の離散化を行うと，

$$\mathbf{Q}_i^{n+1} - \mathbf{Q}_i^n = -\frac{\Delta t}{V_i}\left[\sum_{j(i)} \Delta S_{ij} \mathbf{f}(\mathbf{Q})_{ij}^{n+1} - \sum_{j(i)} \Delta S_{ij} \mathbf{g}(\mathbf{Q})_{ij}^{n+1}\right] \tag{3.23}$$

$$\mathbf{f}(\mathbf{Q})_{ij} = \mathbf{F}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij}, \quad \mathbf{g}(\mathbf{Q})_{ij} = \mathrm{Re}^{-1}\mathbf{G}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij}$$

ここでΔtは時間刻み幅を，添え字nは時間ステップを表わしている．この式中で$\mathbf{Q}^n$を与えて$\mathbf{Q}^{n+1}$を求めるが，$\mathbf{f}$，$\mathbf{g}$は$\mathbf{Q}$の非線形関数であるので，これらの項をテイラー展開して二次導関数以降を省略し，近似的な線形化を施す．その際ヤコビアン$\mathbf{A} = \partial\mathbf{f}/\partial\mathbf{Q}$，$\mathbf{M} = \partial\mathbf{g}/\partial\mathbf{Q}$を用いる．

$$\mathbf{f}(\mathbf{Q})^{n+1} = \mathbf{f}(\mathbf{Q})^n + \mathbf{A}^n\Delta\mathbf{Q}^n, \quad \mathbf{g}(\mathbf{Q})^{n+1} = \mathbf{g}(\mathbf{Q})^n + \mathbf{M}^n\Delta\mathbf{Q}^n \tag{3.24}$$

$$\Delta\mathbf{Q}^n = \mathbf{Q}^{n+1} - \mathbf{Q}^n$$

さらに数値流束ベクトル$\mathbf{f}$には風上差分と同様の取り扱いをすると，節点$i-j$間に関する$n+1$時刻での数値流束ベクトルは次のようになる．

$$\mathbf{f}_{ij}^{n+1} = \mathbf{f}_{ij}^n + \left(\mathbf{A}_i^+\Delta\mathbf{Q}_i + \mathbf{A}_j^-\Delta\mathbf{Q}_j\right)^n \tag{3.25}$$

ここで$\mathbf{A}^{\pm}$は$\mathbf{A}$の正の固有値を持つ行列と負の固有値を持つ行列に分割したものである．これらを式(3.23)に代入して整理し，かつヤコビアン$\mathbf{M}$を省略すると，

$$\left[\frac{V_i}{\Delta t}\mathbf{I}+\sum_{j(i)}\Delta S_{ij}\mathbf{A}_i^{+}\right]\Delta\mathbf{Q}_i+\sum_{j(i)}\Delta S_{ij}\mathbf{A}_j^{-}\Delta\mathbf{Q}_j=\mathbf{R}_i \tag{3.26}$$

$$\mathbf{R}_i=-\sum_{j(i)}\Delta S_{ij}\left(\mathbf{f}-\mathbf{g}\right)_{ij}^{n}$$

となる．さらに式(3.26)の左辺第二項の $\sum_{j(i)}$ を $\sum_{j\in L(i)}$ と $\sum_{j\in U(i)}$ に分割する．

$$\left[\frac{V_i}{\Delta t}\mathbf{I}+\sum_{j(i)}\Delta S_{ij}\mathbf{A}_i^{+}\right]\Delta\mathbf{Q}_i+\sum_{j\in L(i)}\Delta S_{ij}\mathbf{A}_j^{-}\Delta\mathbf{Q}_j+\sum_{j\in U(i)}\Delta S_{ij}\mathbf{A}_j^{-}\Delta\mathbf{Q}_j=\mathbf{R}_i \tag{3.27}$$

この式に LU-SGS 法を適用して $\Delta\mathbf{Q}$ を以下のように二段階に分けて求める．

$$\text{第一段階：}\ \Delta\mathbf{Q}_i^{*}=\mathbf{D}^{-1}\left[\mathbf{R}_i-\sum\Delta S_{ij}\mathbf{A}_j^{-}\Delta\mathbf{Q}_j^{*}\right]$$

$$\text{第二段階：}\ \Delta\mathbf{Q}_i=\Delta\mathbf{Q}_i^{*}-\mathbf{D}^{-1}\sum_{j(i)}\Delta S_{ij}\mathbf{A}_j^{-}\Delta\mathbf{Q}_j \tag{3.28}$$

$$\mathbf{D}=\frac{V_i}{\Delta t}\mathbf{I}+\sum_{j(i)}\Delta S_{ij}\mathbf{A}_i^{+}$$

行列 $\mathbf{D}$ は普通は対角行列でないため，$\mathbf{D}^{-1}$ の演算量は少なくない．そこで $\mathbf{D}$ を近似的に対角行列にして $\mathbf{D}^{-1}$ をスカラー演算に軽減する．そのためにここでは Jameson と Turkel によって提案されたヤコビ行列 $\mathbf{A}$ の分割法[5]を用いる．

$$\mathbf{A}^{\pm}=\frac{\left(\mathbf{A}\pm\rho_A\mathbf{I}\right)}{2} \tag{3.29}$$

ここで ρ_A は $\mathbf{A}$ のスペクトル半径 $|U|+a$ に，粘性流束ベクトルのヤコビアン $\mathbf{M}$ を省略した代わりの項を付け加えたもので，

$$\rho_A=\chi\left(|U|+a\right)+2\frac{\mu}{\mathrm{Re}\cdot\rho\cdot h} \tag{3.30}$$

となる．ここで h は節点 $i-j$ 間の距離で，χ は経験上 1.01 の値をとる．したがって，ヤコビ行列の性質 $\left(\sum_{j(i)}\Delta S_{ij}\mathbf{A}=0\right)$[6]から $\mathbf{D}$ は対角化され，

$$\mathbf{D}=\left[\frac{V_i}{\Delta t}+\frac{1}{2}\sum_{j(i)}\Delta S_{ij}\rho_A\right]\mathbf{I} \tag{3.31}$$

となる．また，近似的に $\mathbf{A}\Delta\mathbf{Q}\approx\mathbf{f}\left(\mathbf{Q}+\Delta\mathbf{Q}\right)-\mathbf{f}\left(\mathbf{Q}\right)=\Delta\mathbf{f}$ としてヤコビ行列の演算を省き，かつ式(3.29)を代入して整理すると，式(3.28)は最終的に以下のようになる．

$$\text{前進スイープ：}\ \Delta\mathbf{Q}_i^{*}=\mathbf{D}^{-1}\left[\mathbf{R}_i-\frac{1}{2}\sum_{j\in L(i)}\Delta S_{ij}\left(\Delta\mathbf{f}_j^{*}-\rho_A\Delta\mathbf{Q}_j^{*}\right)\right]$$

後退スイープ：$$\Delta\mathbf{Q}_i = \Delta\mathbf{Q}_i^* - \frac{\mathbf{D}^{-1}}{2}\sum_{j\in U(i)}\Delta S_{ij}\left(\Delta\mathbf{f}_j - \rho_A\Delta\mathbf{Q}_j\right) \tag{3.32}$$

$$\Delta\mathbf{f}^* = \mathbf{f}\left(\mathbf{Q}+\Delta\mathbf{Q}^*\right)-\mathbf{f}(\mathbf{Q})$$

非構造格子上で LU-SGS 解法を実現するのに最も重要な点は，節点iに隣接する節点jのグループを$L(i)$と$U(i)$，つまり下三角形要素と上三角形要素に分割する方法である．構造格子では，LU-SGS の演算は格子点の番号をi,j,kとすると，$i+j+k=$一定のハイパー面上の点に対して同時に行われるが，非構造格子では格子線がないため，スイープを行うためには格子節点の並び替えによって，ハイパー面を構成しなければならない．ベクトル演算を可能にするためには，一つのハイパー面内の節点は同じハイパー面内の他の節点とつながっていることのないようにしなければならない．また，一つのハイパー面の節点の大多数は，小さな番号を持つハイパー面内の節点と大きな番号を持つハイパー面内の節点の両方からの接続があるようにして，下三角形要素と上三角形要素のバランスを良くする(図3.5)．並び替え手法の詳細は文献[4]に書かれている．

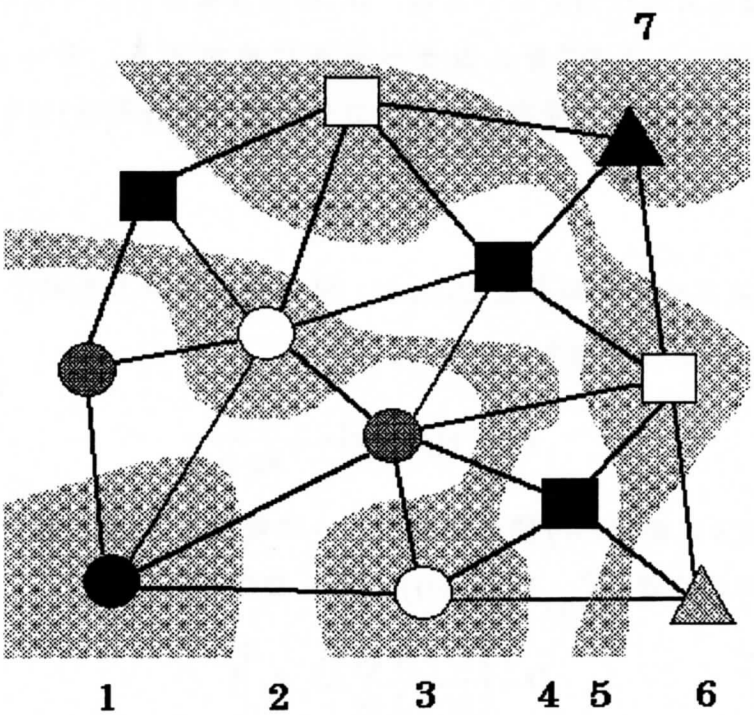


図 3.5 非構造 LU-SGS スイープのためのハイパー面構成

LU-SGS 陰解法による計算では時間精度は維持されない．そこで非定常計算を行う場合，Crank-Nicolson 法を用いた Newton 反復法[7,8]を使用し内部反復を行う

ことにより，線形化や近似因子化による誤差を減らし時間精度を維持する．式(3.32)は Newton 反復法を組み込むことによって次式のように表され，各時間ステップ毎に収束解を求めることができる．

$$\Delta\mathbf{Q}_i^{*(m)} = \mathbf{D}^{-1}[RHS_i^{(m)} - 0.5\,\theta \sum_{j\in L(i)} \Delta S_{ij}(\Delta\mathbf{h}_j^{*(m)} - \rho_A \Delta\mathbf{Q}_j^{*(m)})] \tag{3.33a}$$

$$\Delta\mathbf{Q}_i^{(m)} = \Delta\mathbf{Q}_i^{*(m)} - 0.5\,\theta\,\mathbf{D}^{-1} \sum_{j\in L(i)} \Delta S_{ij}(\Delta\mathbf{h}_j^{(m)} - \rho_A \Delta\mathbf{Q}_j^{(m)}) \tag{3.33b}$$

ここで，

$$\Delta\mathbf{Q}^{(m)} = \mathbf{Q}^{(m+1)} - \mathbf{Q}^{(m)} \tag{3.34}$$

$$RHS^{(m)} = -\left(\mathbf{Q}^{(m)} - \mathbf{Q}^{n}\right) - \Delta t \cdot L^{*}\left(\theta\,\mathbf{Q}^{(m)} + (1-\theta)\mathbf{Q}^{n}\right) \tag{3.35}$$

$\theta = 1/2$ のとき Crank-Nicolson 法となり，時間精度は 2 次精度になる．ここで L^* は，支配方程式をベクトル型で表した以下の式，

$$\frac{\partial\mathbf{Q}}{\partial t} + L(\mathbf{Q}) \equiv \frac{\partial\mathbf{Q}}{\partial t} + \frac{\partial\mathbf{F}_i}{\partial\mathbf{x}_i} + \frac{\partial\mathbf{G}_i}{\partial\mathbf{x}_i} = 0 \tag{3.36}$$

における L の差分演算子を意味し，m は Newton 反復の回数を表している．m は Newton 反復の回数を表している．ここで，式（3.33）～(3.35)において $m=0$ のとき $\mathbf{Q}^{(m)} = \mathbf{Q}^{n}$，$m \to \infty$ のとき $\Delta\mathbf{Q}^{(m)} \to 0$ より $\mathbf{Q}^{(m)} = \mathbf{Q}^{n+1}$ となる．この計算方法では，各時間ステップ毎に 3～5 回の Newton 反復で十分に収束する．

3－4 非圧縮流れへの拡張

従来，圧縮性流れと非圧縮性流れの問題は異なったアルゴリズムで計算されてきた．圧縮性流れに対する解法は時間発展的であり非常に効率的であるが，低速流れに対して適用されると硬直性が現れるという欠点があった．これは，音速と流れの速度に対する対流時間スケールの大きな違いのためである．そのため低速の非圧縮性流れにはふつう，圧力場に対する独立した方程式，すなわち Poisson 方程式を解く方法が用いられてきた．しかし，Poisson 方程式は時間発展形でないため定常計算においては効率的ではなかった．

これらの解法の欠点を補うべく，Chorin[9] は非圧縮性流れに対する時間発展形解法を提案した．これが擬似圧縮性法である．現在では，この擬似圧縮性法によって多くの有用な研究が行われている[10,11]．

非圧縮粘性流体の流れで，外力がある場合には無次元化を行ったときにレイノ

ルズ数以外のパラメータが現れる．非圧縮性の熱流体を例にとると，流体の温度が変化すれば密度も変化する．しかし，非圧縮性流体であるので密度変化は考慮されない．そこで，温度（密度）変化が方程式に及ぼす影響が密度変化によって生ずる外力（浮力）を通してのみ現れ，他の物性値は温度により影響を受けないと仮定する．この仮定をブシネスク近似という．このとき鉛直方向の方程式には浮力による外力項が加わる．また，圧縮性流体を取り扱うときには基礎方程式にエネルギー方程式が含まれており，自動的に熱流体の取り扱いができるが，非圧縮性を仮定した場合には，熱流体の方程式が必要となる．

3 次元非圧縮 Navier-Stokes 方程式は，擬似圧縮性法を導入することにより 式(3.1)の積分形で表した式と同じになる．つまり，

$$\frac{\partial}{\partial t}\iiint \mathbf{Q}dV + \iint \mathbf{F}\cdot\mathbf{n}\,dS - \iint \mathbf{G}\cdot\mathbf{n}\,dS = \iiint \mathbf{S}dV \tag{3.37}$$

ここに，**Q** は変数ベクトル，**F** は非粘性数値流束ベクトル，**G** は粘性流束ベクトル，**S** は外力ベクトルであり，**Q**，**F**，**G** は以下のようになる．

$$\mathbf{Q} = \begin{Bmatrix} p \\ u \\ v \\ w \\ T \end{Bmatrix}, \tag{3.38}$$

$$\mathbf{F}\cdot\mathbf{n} = \begin{Bmatrix} \beta\Theta \\ \Theta u + pn_x \\ \Theta v + pn_y \\ \Theta w + pn_z \\ \Theta T \end{Bmatrix}, \left(\Theta = \mathbf{v}\cdot\mathbf{n}\right) \tag{3.39}$$

$$\mathbf{G}\cdot\mathbf{n} = \frac{1}{\mathrm{Re}}\begin{Bmatrix} 0 \\ \tau_{xx}n_x + \tau_{xy}n_y + \tau_{xz}n_z \\ \tau_{yx}n_x + \tau_{yy}n_y + \tau_{yz}n_z \\ \tau_{zx}n_x + \tau_{zy}n_y + \tau_{zz}n_z \\ \left(\nabla T\cdot\mathbf{n}\right)/\mathrm{Pr} \end{Bmatrix} \tag{3.40}$$

$\mathbf{v} = \{u, v, w\}$，p および $\mathbf{n} = \{n_x, n_y, n_z\}$は速度ベクトル，圧力および x, y, z 方向の単位法線ベクトルである．β は擬似圧縮性パラメータ，Re はレイノルズ数，Pr は

プラントル数である．

流束の計算は圧縮性流れ同様，検査体積の表面に沿った点を用いて各検査体積について行われる．式(3.14)において，$\mathbf{Q}_L, \mathbf{Q}_R$はその辺について検査体積境界面の両側における保存変数ベクトルの値であり，流束のヤコビ行列$\mathbf{A}=\frac{\partial \mathbf{h}}{\partial \mathbf{Q}}$は疑似圧縮性近似により以下のようになる．

$$\mathbf{A}=\begin{bmatrix} 0 & \beta n_x & \beta n_y & \beta n_z & 0 \\ n_x & \Theta+un_x & un_y & un_z & 0 \\ n_y & vn_x & \Theta+vn_y & vn_z & 0 \\ n_z & wn_x & wn_y & \Theta+wn_z & 0 \\ 0 & Tn_x & Tn_y & Tn_z & \Theta \end{bmatrix}, \tag{3.41}$$

ここで，

$$|\mathbf{A}|=\mathbf{R}|\Lambda|\mathbf{L} \tag{3.42}$$

であり，$|\Lambda|$は流束行列$\mathbf{A}$の固有値を要素に持つ対角行列である．各固有値は以下のように与えられる．

$$\begin{aligned} &\{\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5\}=\{\Theta-C,\Theta+C,\Theta,\Theta,\Theta\} \\ &C=\sqrt{\Theta^2+\beta} \end{aligned} \tag{3.43}$$

流体中に温度差（密度差）が生ずると，鉛直上向きに浮力が働く．密度変化と温度変化の間には，以下のような関係が仮定できる．

$$\rho-\tilde{\rho}=-\rho\delta\left(T-\tilde{T}\right) \tag{3.44}$$

ここでδは体積膨張率とよばれ，$\delta=-\ln\rho/\partial T$で定義される．これにより鉛直方向には，

$$F_b=\delta g\left(T-\tilde{T}\right) \tag{3.45}$$

なる浮力F_bが加わる．浮力F_bは以下のように無次元化される．

$$F_b'=\frac{\mathrm{Ra}}{\mathrm{Re}^2\cdot\mathrm{Pr}}T \tag{3.46}$$

ここで，Raはレイリー数とよばれる対流の強さを示す無次元数で，粘性力と浮力との比で表される．

回転する地球上の大気や海洋の流れなどのように，剛体回転からのずれが比較的小さい流れは，非回転流体に見られない特有の性質を持っている．このような

運動を記述するには，回転する座標系を用いるのが便利である．しかし，回転座標系は非慣性系なので，コリオリ力という見かけの力が現れる．流体運動が回転流体としての性質を強く持つかどうかはコリオリ力の役割の大小で決まる．剛体回転の角速度ベクトルを$\dot{\mathbf{U}}$，これに相対的な流体の速度ベクトルを$\mathbf{u}$，とするとコリオリ力ベクトル$\mathbf{F}_c$は以下のように記述される．

$$\mathbf{F}_c = 2\dot{\mathbf{U}} \times \mathbf{u} \tag{3.47}$$

以上の外力を考慮すると，式(3.37)中の外力ベクトル$\mathbf{S}$は以下のように表される．

$$\mathbf{S} = \begin{Bmatrix} 0 \\ f_x \\ f_y + \dfrac{\mathrm{Ra}}{\mathrm{Re}^2 \cdot \mathrm{Pr}} T \\ f_z \\ 0 \end{Bmatrix} \tag{3.48}$$

ここで，f_x, f_y, f_zはそれぞれコリオリ力のx, y, z方向成分である．

参考文献

[1] Obayashi, S. and Guruswamy, G. P., "Convergence Acceleration of an Aeroelastic Navier-Stokes Solver," AIAA Paper 94-2268, 1994.

[2] Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady State Solutions," AIAA Paper 93-0880, 1993.

[3] Parthasarathy, V. and Kallinderis, Y., "Adaptive Prismatic- Tetrahedral Grid Refinement and Redistribution for Viscous Flow," *AIAA Journal.*, Vol.34., No.4, pp.707-716, 1996.

[4] Sharov, D. and Nakahashi, K., "Reordering of Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations," *AIAA Journal.*, Vol.36, No.3, pp.484-486, 1998.

[5] Jameson, A. and Turkel, E., "Implicit Scheme and LU Decompositions," *Mathmatics of Computations*, Vol.37, No.156, pp.385-397, 1981.

[6] Men'shov, I. and Nakamura, Y., "Implementation of the LU-SGS Method for an Arbitrary Finite Volume Descretization," *Proc. of Japanese 9th CFD Symposium*, 1995, pp.123-124.

[7] Rai, M. M., "Unsteady Three-Dimensional Navier-Stokes Simulations of Turbine Rotor-Stator Interaction," AIAA Paper 87-2058, 1987.

[8] Kano, S., Yamamoto, S., and Daiguji, H., "An Efficient CFD Approach for Simulating Unsteady Hypersonic Shock-Shock Interference Flows," Computational Fluid Dynamics, Vol. 2, 1995, pp. 571-574.

[9] Chorin, A. J. (1967): A Numerical Method for Solving Incompressible Viscous Flow Problems,
J. of Comput. Physics, Vol. 2, pp. 12-26.

[10] Yoon, S. and Kwak, D. (1991): Three-Dimensional Incompressible Navier-Stokes Solver Using Lower-Upper Symmetric-Gauss-Seidel Algorithm, *AIAA J.*, Vol. 29, pp.874-875.

[11] Rogers, S. E., Kwak, D. and Kiris, C. (1991): Steady and Unsteady Solutions of the Incompressible Navier-Stokes Equations, *AIAA J.*, Vol. 29, pp.603-610.

第４章　非構造格子ＣＦＤの検証および応用と展開

4-1 試験検証問題

第２章，第３章において記述した非構造 CFD を主に(1) デルタ翼[1,2]，(2) スクラムジェットエンジン[3-11]，(3) 超音速実験機[15-17] の問題で検証するとともに，更に表面格子生成法の改良と提案[13,14,18]，Navier-Stokes 計算のためのハイブリッド非構造格子生成法の提案[19] をおこなった．

また第３章で述べたソルバーについては，陰解法の開発[20]と高レイノルズ数流れでの検証[21]，マルチグリッド法の提案[22]，空間前進解法の提案[23,24]等の高速化計算手法の開発を進めた．また，適用範囲の拡張として非圧縮流れへの計算コード[25-27]を開発した．

更に，流体機械最適化の研究としては，非構造格子と逆解法の組み合わせによる超音速実験機主翼の設計[28]，およびAdjoint 方程式法による設計法[29]を提案している．

研究成果の詳細は添付論文に譲り，ここでは検証問題についての結果を以下に簡単に説明する．

4-2 デルタ翼

デルタ翼は古くから実験的研究が行われてきており，また数値シミュレーション研究の課題としても多用されており，本非構造 CFD の検証課題として適しているといえよう．また，大きな迎え角では，デルタ翼は前縁剥離渦を翼上面に形成するが，この縦渦は翼面からは離れているため，従来の構造格子 CFD では十分に渦を解像することが困難であった．この点，非構造格子は任意の領域の格子を細かくすることが可能であり，その効果を試験するためにも高迎角デルタ翼は適した計算対象である．

デルタ翼の計算では，まず非構造ハイブリッド格子による計算手法がどこまで精度良い計算が行えるかのテストを行った．結果としては，従来の構造格子ＣＦＤと同等の精度，および計算時間で可能であることが示された．また流れ場の密度勾配をモニターしての解適合格子細分化で精度向上が行えることを示し，この種の問題に対する非構造 CFD の優位性を示した．この研究内容は，それに付随する研究成果とともに文献[1]に詳述されている．

デルタ翼の前縁剥離渦は，更に大きな迎え角では渦崩壊という現象を伴う．この現象の精度良い数値シミュレーションには更に高解像な格子を必要とし，解適合格子細分化が必須の要素となる．しかしながら，この種の渦運動に対しては，どの領域に格子細分化を行えばもっとも効果的であるかは難しく，流れ場のモニター量を決めるには試行錯誤で行われていたのが現状である．このような解適合細分化の難しさを解決するため，流れ場内における流れの特徴線を用いた新しい解適合手法を提案した．つまり，高迎角デルタ翼で現れる縦渦に対し，その渦中心を同定してその領域に格子細分化を行うという手法である．このような流れ特徴線という幾何学量を解適合細分化のモニター量として使うことにより，従来の方法での曖昧さを除去でき，かつ効率よい細分化手法を構築できた．実際，本手法による数値シミュレーション結果は，渦崩壊の精度良い再現を可能にしている．この研究の詳細は文献[2]に詳述されている．

4-3 スクラムジェットエンジン

スクラムジェットエンジンは，超音速・極超音速飛行用のエンジンとして開発が国内外において進められているが，流体力学的には様々な課題を抱えており，その数値解析研究は非常に重要である．エンジンの要素としては，超音速流れを減速圧縮するインテークと，その下流部の燃焼器部，

そして燃焼流れを膨張させるノズル部とからなる．ここでは航空宇宙技術研究所で開発されているスクラムジェットエンジンに対し，非構造 CFD による数値解析を行い，実験値との比較検証を進めた．

文献[3-6]はスクラムジェットエンジンのインテーク部の数値解析研究であり，非構造 CFD の精度検証を行うとともに，インテーク内の衝撃波と境界層の干渉について論じている．スクラムジェットエンジンの流路形状は比較的単純ではあるものの，鋭利なインテークカウルやストラット等では従来の構造格子 CFD では格子特異点を形成し，その解析を困難にしている．これに対し非構造 CFD では特異点が現れず，そのため計算も安定に行えた．計算精度としても従来の手法に遜色ないレベルであることを示し，非構造 CFD に対する従来の悪いイメージを払拭した．

スクラムジェットエンジンへの非構造 CFD の適用は，さらに航空宇宙技術研究所モデル全体について行い，壁面摩擦抵抗等を実験データと詳細に突き合わせて計算の信頼性と実験データ解析をおこなった[7,8]．また，スクラムジェットエンジンの最大の課題は超音速燃焼であり，まず燃焼器部の解析結果から着火限界の議論を行う[9]．そして更にソルバーに燃焼素反応をとりいれた拡張を行い，燃料噴射穴近傍の燃焼流シミュレーションまでを可能にした[10,11]．

4-4 超音速実験機

現在，航空宇宙技術研究所（NAL）が中心となって次世代超音速旅客機の研究が精力的に行われており，小型超音速実験機を用いた飛行試験が 2002 年に計画されている[12]．この実験では，エンジンなしの超音速実験機に固体ロケットブースターを取り付けて打ち上げ，高度 15,000m，マッハ数 2.5 に到達した時点でブースターを切り離し，その後機体を滑空させて揚力・抗力や機体表面圧力などの空力設計に必要なデータを取得する．

ロケットブースターを用いて超音速実験機を打ち上げる際，分離時の空力干渉はもちろんのこと遷音速から超音速域での空力性能を正確に評価する必要がある．すでに風洞試験が NAL において実施されており，ここでは CFD により空力評価を行い，実験と対比するとともに実験を補うデータ収集を目的としている．また，この種の複雑な形状については CFD アルゴリズム研究の点からも興味ある課題であり，CFD の実用性の検証にもなろう．

(1) 打ち上げ形態の詳細計算

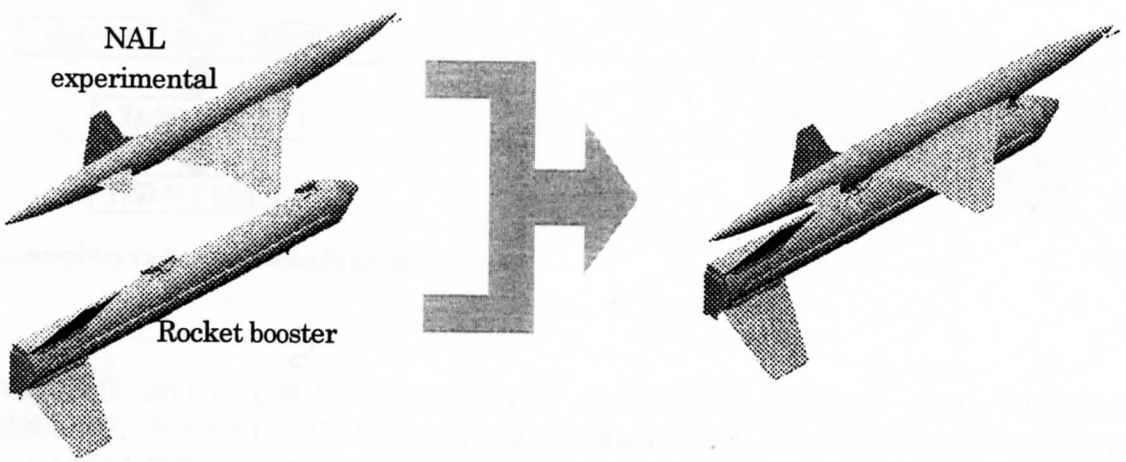


Fig. 4.1 Superposition of two CAD models

超音速実験機の打ち上げ形態の CAD 出力を図 4.1 に示す．これは航空宇宙研究所で作成された CAD

モデルであり，機体にはピトー管，固体ロケットブースターには燃料配管のフェアリングや打ち上げ時に必要な金具，また両者の結合部の部品などが実機どおりほぼ正確に表現されている．この周りの流れを計算する場合，CAD からの出力データをもとに表面格子を生成し，更に外部境界の設定と空間格子生成が必要となる．この計算前処理において，CAD 出力から表面格子生成までは CFD 計算の中で最も時間を費やす部分であるとともに，計算精度にも大きな影響を及ぼす．

今回扱う打ち上げ形状では，別々にモデリングされたロケットブースターと超音速実験機を，当研究室にある CAD ソフト・CATIA を利用して 1 つのモデルにまとめた（図 4.1）．この際，CFD 計算を行うためには CAD モデルそのものについて数ヶ所の修正を行った．1 つは実験機頭部につけられたピトー管の静圧穴であり，全体の流れに影響ないため埋める作業をした．また，実験機とブースターの接合部も非常に狭い隙間が存在する．このような隙間は限られた計算機容量では再現が困難であり，かつ流れへの影響も無視できる程狭いことから，CAD 上で埋める修正を施した．

このモデルに対し表面格子生成を行うため，CAD から STL (Stereolithography) データを出力した．このデータは，CAD 出力形式の 1 つであり光造形用出力形式とも呼ばれる．3 次元自由曲面を三角形面の集合体として近似する方式で，CAD から造形装置にデータを渡す場合に用いられる一般的な出力書式である．

この STL 出力を基に表面格子生成を行う．本研究では伊藤ら[13, 14]により開発された表面格子生成プログラム Edge Editor を利用した．このソフトウェアでの格子生成手順を図 4. 3 に示す．最初に形状データを STL 形式として読み込んだ後，表面格子生成に必要となる背面格子を生成するための前処理が行われる．その後この背面格子に対してアドバンシング・フロント法が適用される．空間格子生成に必要となる外部・対称面境界格子も同様に作成される．

このように定義された閉領域に四面体分割を施し，空間格子を生成する．ここでは，Delaunay 分割法に基づく手法を用いたエラー! **参照元が見つかりません。**．これにより得られた空間格子生成の総節点数は 658, 532 点，四面体要素数は 3, 570, 327 個であった．

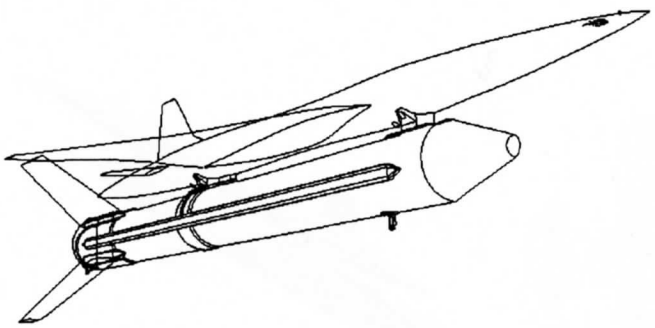


Fig. 4.2 Reconstruction of geometric features

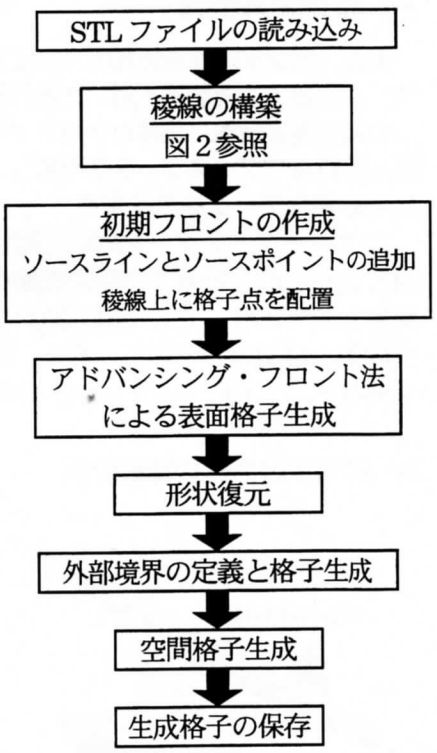


Fig. 4.3 Flowchart of surface grid generation

非構造格子ソルバーにより Euler 計算を行った．図 4.4 に自由流マッハ数 M_∞= 1.05，迎角 0 度の計算条件における等圧力線図を示す．同様に図 4.6 には M_∞= 2.0 における圧力分布を示す．SST 実験機に取り付けてあるピトー管や固体ロケットブースターと実験機の結合部部品で生じた衝撃波がきちんと捉えられ，それらが機体やロケットブースターと干渉している．またロケットブースター先端で発生した衝撃波が実験機下面で反射して再びロケットブースターと干渉している様子も分かる．

また，ピトー管や取り付け金具などの影響を調べるため，それら詳細部品を CAD 上で取り除いた形

状についてもCFD計算を行った．図4.5に自由流マッハ数M_∞= 1.05，迎角0度における機体表面圧力分布と対称面等圧力線図を記す．

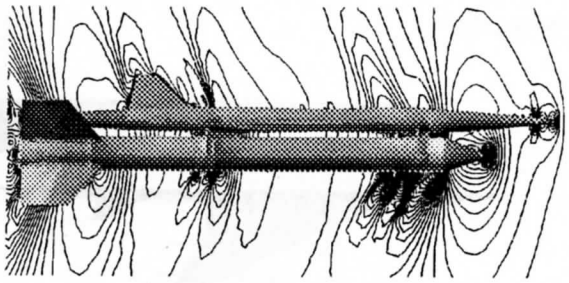


Fig. 1.4 Computed pressure contours at Mach 1.05, α=0deg

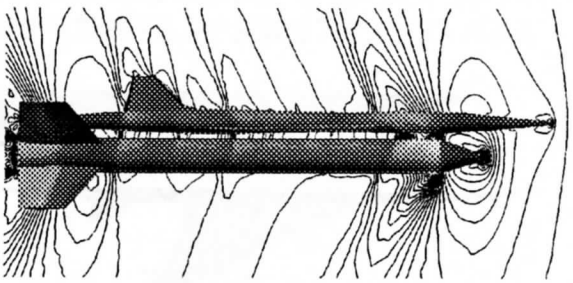
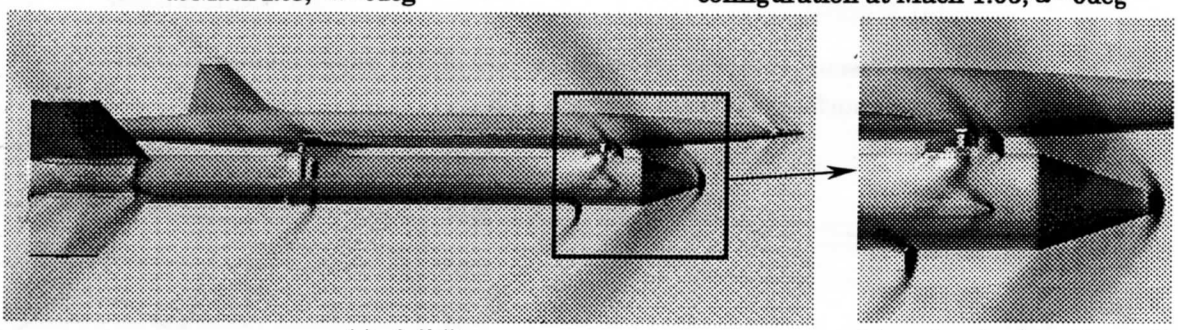


Fig. 4.5 Computed pressure contours of smooth configuration at Mach 1.05, α=0deg



(a) 全体像　　(b) ロケットブースター先端

Fig. 4.6 Computed pressure contours at Mach 2.0, α = 0deg

風洞試験の結果と比較するために，詳細な打ち上げ形状と突起物を取り除いた形状について，計算結果から揚力係数C_Lと抗力係数C_Dを算出した（図4.7）．突起物は，特に従来の構造格子CFDでは再現が困難であるため省かれることが多いが，ここではそれら詳細部品の空力性能への影響を調べることを目的としている．また，実験値は詳細な打ち上げ形状についての結果である．本計算は粘性を無視したEuler計算であるため，風洞試験で得られたC_Dから風洞試験レイノルズ数における摩擦抵抗推定値をそれぞれ引いた値を使用している．詳細形状のC_Lについては，マッハ数0.9，1.05では実験値と計算値で約10%程のずれがみられる．これは遷音速域での衝撃波と境界層の干渉のためと思われ，この速度域ではNavier Stokes計算が必要であろう．その他は6%以下の誤差であり，非粘性計算であるにも関わらずほぼ良好な結果が得られている．今後さらに格子点数を増やした計算を行えば，より精度よい結果が得られるだろう．また，C_Dについては，同じく衝撃波・境界層干渉の影響が無視できないM_∞= 0.9では実験値と計算値で約35%という大きな食い違いが見られるが，その他の条件では計算値が10%程度実験値を上回る値で同じ傾向を示している．

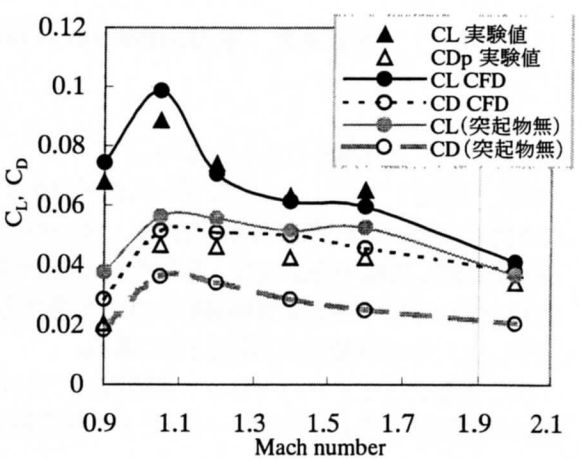


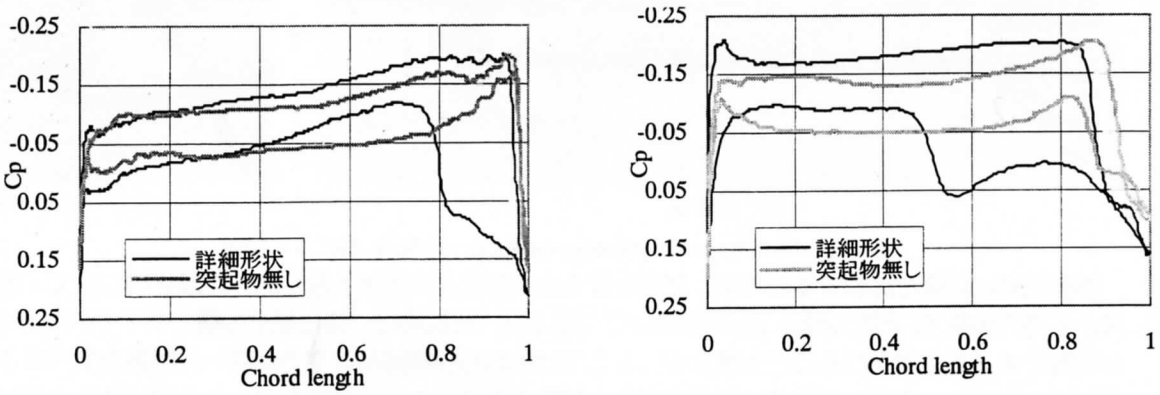
Fig. 4.7 Lift and drag coefficients

次に実験機の詳細形状と突起物無し形状について比較してみると，特に遷音速付近でC_Lに関して50%前後の違いが認められる．この理由として，遷音速域ではロケットブースターの取り付け金具の存在で，その少し上流に衝撃波が発生し，それが実験機の主翼下面の圧力分布を大幅に変えていることが想像される．実際，図4.8に示すように，主翼下面には詳細形状では衝撃波が確認できる．また，図の断面①と断面②について，主翼周りの圧力係数分布の比較を図9に示す．突起物の存在によって，主翼

下面に衝撃波が発生して圧力分布が大きく変化しており，これが図4.7におけるC_Lの大きな相違となっている．これにより，遷音速域での空力評価には，たとえ全体形状に対して非常に小さな突起物でも，全体のC_Lに大きく影響を与える可能性があることが分かる．

(a) 詳細形状　　(b) 突起物無し形状

Fig. 4.8 Pressure contours on under surface of SST experimental airplane and upper surface of booster at Mach 1.05



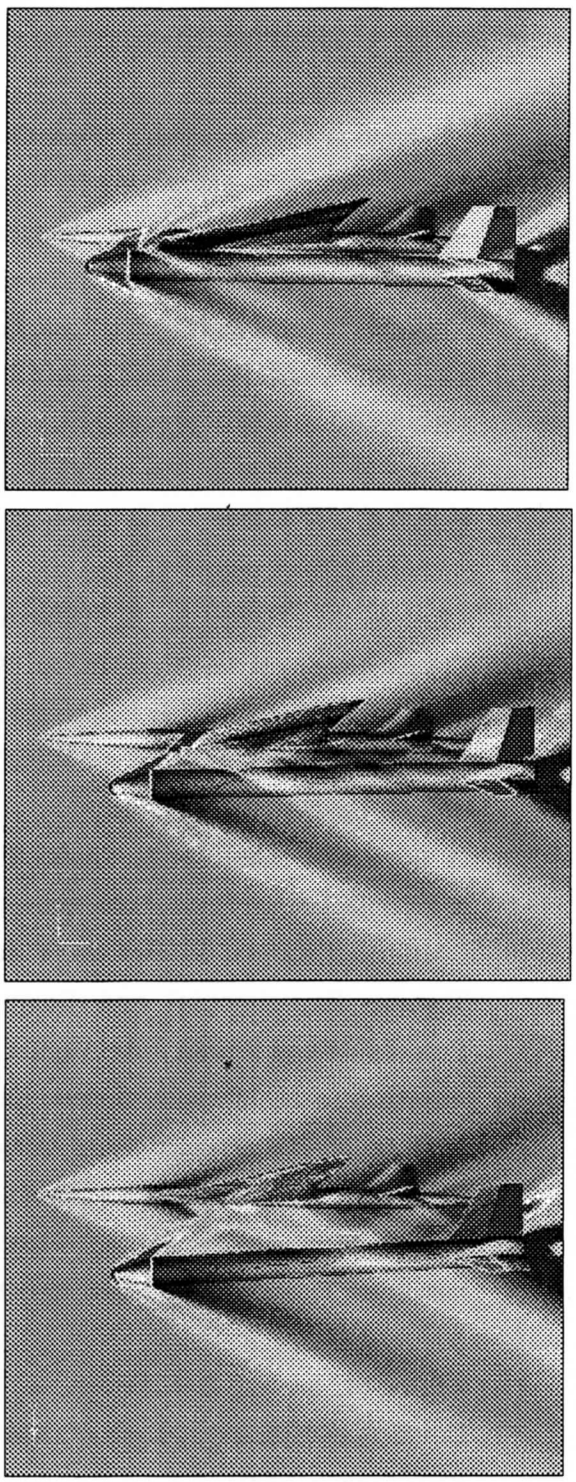
(a) 断面①　　(b) 断面②

Fig. 4.9　Cp distribution at main wing at M=1.05

(2) ブースター分離シミュレーション

次に，高度１５０００mでのブースターロケットと実験機の分離シミュレーションを行う．この相互に移動する物体を非構造で扱うために，オーバーセット非構造格子法を新たに提案した．これは，個々の物体に対して格子を生成し，それぞれの格子が相互に動くことを許す方法である．これを容易にかつ確実に行うための格子間情報構築の方法をあたらに提案[15]し，正確な形状によるシミュレーションを実行した．その詳細は文献[16,17]に譲る．

ここで提案したオーバーセット非構造格子法は，非構造格子の適用範囲を容易に拡大するとともに，従来の構造格子を用いたオーバーセット法の問題点を解決する方法として，非常に注目されている．



参考文献（*は添付論文）

[1]* S. Kano and K. Nakahashi,, "Flow Computations around Delta Wings Using Unstructured Hybrid Grids", Journal of Aircraft, Vol.36, No.2, pp.374-379, 1999.

[2]* Murayama, M., Nakahashi, K., Sawada, K., "Numerical Simulation of Vortex Breakdown Using Adaptive Refinement with Vortex-Center Identification", AIAA-2000-0806, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.

[3] 小寺正敏，中橋和博，大林茂，苅田丈士，三谷徹，"スクラムジェットインレット内における境界層流入の影響，"日本航空宇宙学会誌，Vol. 45，No. 519，平成９年４月，pp. 216-221.

[4]* 小寺正敏，中橋和博，Dmitri Sharov，"ハイブリッド非構造格子法による３次元衝撃波／乱流境界層干渉の数値シミュレーション，" 日本機械学会論文集(B 編)，64 巻，627 号，pp.3669-3674, 1998 年 11 月.

[5]* Kodera, M., Nakahashi, K., Hiraiwa, T., Kanda, T., and Mitani, T., ""Scramjet Inlet Flow Computations by Hybrid Grid Method," AIAA Paper 98-0962, 1998.

[6]* 小寺正敏，中橋和博，五十嵐康隆，苅田丈士，平岩徹夫，三谷徹，"ハイブリッド非構造格子法を用いたスクラムジェット内部流の数値解析"，日本機会学会論文集（B編）６５巻,６３３号, pp.1513-1519, 1999.

[7]* Y. Igarashi, K. Nakahashi, M. Kodera, T. Mitani, T. shimura, ""Experimental and Numerical Analysis of Scramjet Internal Flows -Comparative Studies of Engine Drag-", AIAA 98-1512, April 1998.

[8] T. Mitani, T.Kanda, T. Hiraiwa, Y. Igarashi, K. Nakahashi, "Drags in Scramjet Engine Testing: Experimental and Computational Fluid Dynamics Studies", Journal of Propulsion and Power, July-August 1999. Vol.15, No.4.

[9] Yasutaka Igarashi, Masatoshi Kodera, Kazuhiro Nakahashi and Tohru Mitani, "Computation of cold flows inside of a scramjet engine using unstructured grid" 1st Eastern-Western High Speed Flow Fields Conference and Workshop, Kyoto, Nov. 1998.

[10] Kodera, M., and Nakahashi, K, "Extension of Unstructured Hybrid Grid Method to Supersonic ombustion Flows" AIAA Paper 99-0486, 37th Aerospace Sciences Meeting and Exhibit, Reno, January 1999.

[11] Kodera, M., Sunami, T., Nakahashi, K., "Numerical Analysis of SCRAMHET Combusting Flows by Unstructured Hybrid Grid Method", AIAA 2000-0886, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.

[12] Iwamiya, T., "NAL SST Project and Aerodynamic Design of Experimental Aircraft", Proc. Of the 4th ECCOMAS Computational Fluid Dynamics Conference, Athens, John Wiley & Sons Ltd., Chichester, 1998, pp.580-585.

[13]* Ito, Y., Nakahashi, K., "Direct Surface Triangulation Using Stereolithography (STL) Data", AIAA-2000-0924, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.

[14] 伊藤 靖，中橋 和博，"GUI を用いた非構造表面格子生成法"，日本航空宇宙学会誌，Vol. 48, No. 554 (3 月号)，2000.

[15]* K. Nakahashi, F. Togashi, D. Sharov, "An Intergrid-Boundary Definition Method for Overset Unstructured Grid Approach", Proc. 14th AIAA Computational Fluid Dynamics Conference, AIAA-99-3304, 1999.

[16]* Togashi, F., Nakahashi, K., Ito, Y., Iwamiya, T., Shimbo, Y., "Flow Simulation of NAL Experimental Supersinic Airplane/Booster Separation Using Overset Unstructured Grids", AIAA-2000-1007, 38th Aerospace Sciences Meeting and Exhibit, Reno, January 2000.

[17] 富樫史弥，中橋和博, "非構造オーバーセット格子による超音速実験機・ブースター分離シミュレーション", 日本航空宇宙学会誌，Vol. 48, No. 556 (5 月号)，2000.

[18]* D. Sharov, and K. Nakahashi, ""Curvature Adapted Triangulation of Surface Models via Incremental Insertion Algorithm", Proc. of 6th International Conference on Numerical Grid Generation in Computational Field Simulations, pp.695-704, 1998.

[19]* D. Sharov, K. Nakahashi, "Hybrid Prismatic/Tetrahedral Grid Generation for Viscous Flow Applications", AIAA Journal, Vol.36, No.2, pp.157-162, 1998.

[20]* D. Sharov, K. Nakahashi, "Reordering of Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations", AIAA Journal, Vol.36, No.3, pp.484-486, 1998.

[21]* K. Nakahashi, D. Sharov, S. Kano, M. Kodera, "Applications of Unstructured Hybrid Grid Method to High-Reynolds Number Viscous Flows ", International Journal for Numerical Methods in Fluids, Vol.31, pp.97-111, 1999.

[22]* N. Okamoto, K. Nakahashi, S. Obayashi, "Unstructured Multigrid Method for Euler Equations", Proc. Int. Conf. on Fluid Eng., JSME Centennial Grand Congress, Tokyo, Vol.2 (1997), pp.801-805.

[23]* K. Nakahashi and E. Saitoh, "Space-Marching Method on Unstructured Grid for Supersonic Flows with Embeded Subonic Regions", AIAA Journal, Vol.35, No.8, pp.1280-1285, August 1997.

[24]* H. Morino, K. Nakahashi, "Space-Marching Method on Unstructured Hybrid Grid for Supersonic Viscous Flows", AIAA Paper 99-0661, 37th Aerospace Sciences Meeting and Exhibit, Reno, January 1999.

[25]* Sharov, D., Nakahashi, K., "Low Speed Preconditioning and LU-SGS Scheme for 3-D Viscous Flow Computations on Unstructured Grids", AIAA Paper 98-0614, 1998.

[26]* 加藤，中橋：非構造格子上での擬似圧縮性法を用いた非圧縮流れ解析の精度検証，日本機械学会論文集 B 編，65 巻 640 号（1999），pp. 3899-3905.

[27]* 加藤，村山，伊藤，中橋：非構造格子上における非定常非圧縮流れの計算 日本機械学会論文集 B 編，66 巻 641 号（2000），pp. 4-5.

[28]* Obayashi, S., Nakahashi, K., Oyama, A. and Yoshino, N., "Design Optimization of Supersonic Wings Using Evolutionary Algorithms," Proceedings of the Fourth ECCOMAS Computational Fluid Dynamics Conference, Athens, John Wiley & Sons Ltd, Chichester, 1998, pp. 575-579.

[29]* H-J. Kim, S. Obayasshi, K. Nakahashi, "Aerodynamic Optimization of Supersonic Wing-Nacelle Configuration Using an Adjoint Method with the Unstructured-Grid Approach", International Workshop on Numerical Simulation Technology for Design of Next Generation Supersonic Civil Transport, Tokyo, Jan. 2000.

第５章　成果のまとめおよび今後の展望

非構造格子に基づく流体計算法の総合的な開発とその検証を行った結果，以下の成果が得られた．

(1) CAD データから幾何学的特徴線を利用した効率よい表面格子生成法の開発
(2) 非構造空間格子生成コードの開発と整備
(3) Euler/Navier-Stokes ソルバーの計算時間短縮法の開発
　(3.1) 陰的時間積分法
　(3.2) マルチグリッド法
　(3.3) 空間前進解法
(4) 非圧縮性流れへの拡張
(5) デルタ翼流れに対する検証と流れの特徴線を利用した解適合細分化法の提案
(6) スクラムジェットエンジンの数値解析とその検証，燃焼流れへの拡張
(7) 超音速実験機の数値解析と検証，
(8) 移動物体に対するオーバーセット非構造格子法の提案とブースター分離のシミュレーション
(9) 非圧縮流れへの拡張
(10)　形状最適化法との組み合わせ

これらの成果は，非構造格子 CFD の工学的有用性を非常に高め，かつ検証問題の結果から計算精度としても十分であることを確認した．また，本研究において新たに提案した計算手法は今後の非構造格子 CFD の更なる拡張を期待させる．

しかし，非構造格子 CFD はまだまだ問題点も抱えている．もっとも大きな問題は Navier-Stokes 計算用の格子生成である．本研究でもハイブリッド格子生成コードを開発したが，必ずしも信頼性があるともいえず，複雑な形状ではハイブリッド格子生成がボトルネックとして存在している．このハイブリッド格子生成は，今後の最重要事項の研究課題である．逆に，ハイブリッド格子が自動的に生成できるようなコードが開発されれば，非構造格子 CFD は従来の構造格子 CFD に完全に取って代わるだけのポテンシャルを持っている．

非構造格子ＣＦＤの別の問題点として，構造格子 CFD に比べてメモリー要求が大きいこともあげられる．通常のプログラミングでは，構造格子 CFD のソルバーに比べ約３倍から４倍程度のメモリーが必要であり，更に解適合細分化等の機能を入れると，メモリー要求は非常に大きくなる．この問題は格子データが非構造であることからの本質的なものであり解決策はない．ただし，今後とも計算機メモリーの低価格化，大規模化が進むことから，さらには今後の分散メモリー並列計算機の利用で，メモリー要求は大きな問題点ともならないであろう．

CFD を真の流体機械設計ツールとして用いられるには，計算時間も一層の短縮が望ましい．陰的時間積分の導入で数年前に比べ１０分の１程に計算時間が短縮されたものの，まだまだ

短縮の余地が残されている．一つにはGMRESの導入であり，またマルチグリッドと陰解法との組み合わせも大幅な効率改善が見込まれる．そして大規模計算での特に大幅な計算時間短縮には，ソルバーの並列化が不可欠である．構造格子上のソルバーは並列化が比較的簡単であるが，非構造格子ソルバーの並列化は，領域分割の方法がもっぱら用いられる．Mavriplisらは，CRAY T3Eの1450プロセッサーを用いて飛行機周りの計算を行い，24.7×10^6格子点の大規模計算でも並列化により工学的に有用な時間範囲内で結果を得ることができることを示した．領域分割を用いることにより非構造格子でも高い並列化効率が得られることを実証しており，今後とも並列計算への対応が進むであろう．

本研究課題を通じて非構造格子CFDの高度化と検証を行い，成果は予想以上に得られた．また，従来の構造格子CFDに非構造格子は勝るとも劣らないことが実証された．今度，益々非構造格子CFDが工学現場に使われていくことであろう．また，そのためにも，今回開発したコードを近い将来にはパブリックドメインソフトとすることも前向きに考えていきたい．

付録（代表的な成果論文）

デルタ翼のNavier-Stokes計算